Tutorial:

The Keil MCB1700 Evaluation Board

By Le Diem Tho

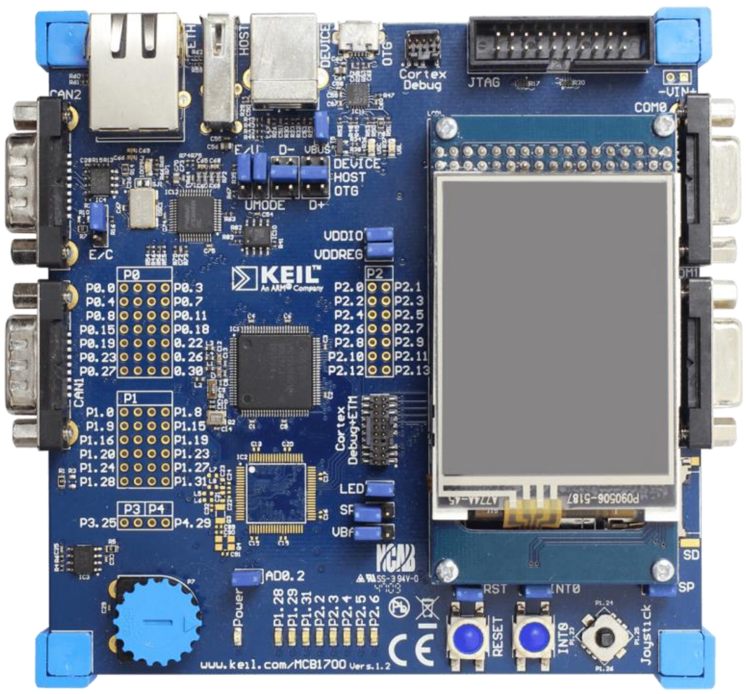


Table of content

Chapter I: Introduction

1.1. Objectives

After completing this lab, you will able to:

- Install Keil-Lite MDK to develop solution for ARM-based microcontrollers.
- Understanding the basic structure of the Keil MCB1700 Evaluation Board
- Understand and be able to control GPIOs.

1.2. Pre-lab Requirement

1.2.1. Installing Keil MDK:

Download and install the Keil-Lite MDK:

http://www.keil.com/mdk5/editions/lite/

1.2.2. Learn about the Keil MCB1700 Evaluation Board

Download "MCB1700 Quick Start Guide":

http://www.keil.com/mcb1700/mcb1700_quickstart.pdf

Download LPC176x User Manual:

http://www.nxp.com/documents/user_manual/UM10360.pdf

1.3. In-lab Requirement

1.3.1. Creating new projects and source files

- Open "Keil uVision". From menu, select *"Project → New uVision Project…"*

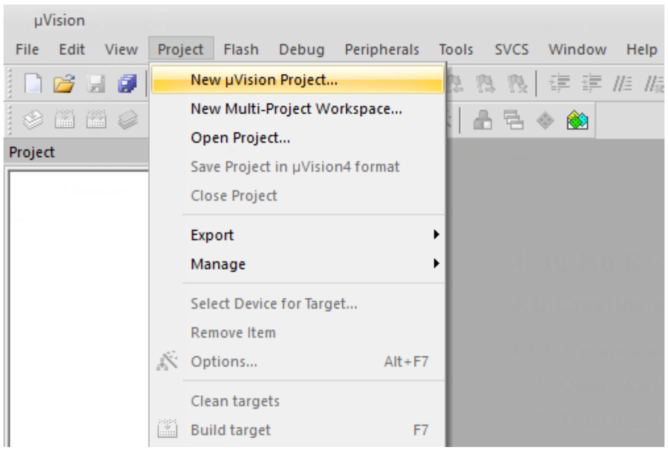
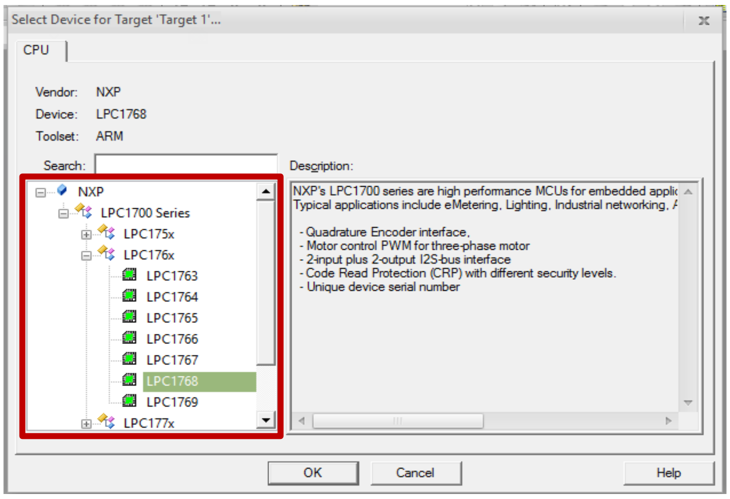


Figure 1: Creating a new project.

- "Create New Project" window appears. In this window, change your project's name to "Lab1_StudentName_StudentID" (Your real name and ID). Then click "Save".
- "Select Device" window appears. In the list box on the left side of the window, search for "LPC1768", which is the microcontroller of our boards.



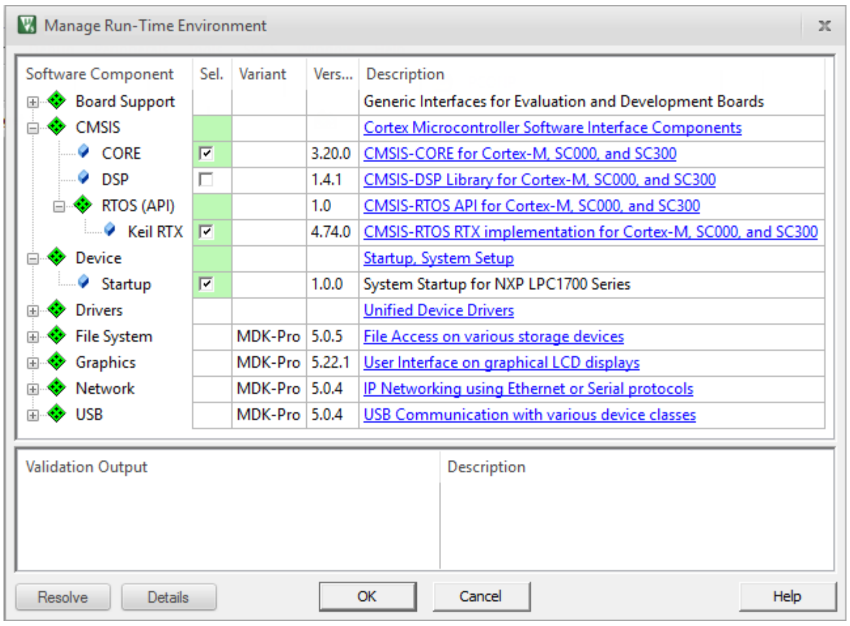
Vendor: NXP
Device: LPC1768
Toolset: ARM

NXP's LPC1700 series are high performance MCUs for embedded applic
Typical applications include eMetering, Lighting, Industrial networking, A

- Quadrature Encoder interface,
- Motor control PWM for three-phase motor
- 2-input plus 2-output I2S-bus interface
- Code Read Protection (CRP) with different security levels.
- Unique device serial number

Figure 2: Select Device for your target.

- "Manage Run-Time Environment" window appears. Select components that we want to include into our program. In this lab, select these components: "CORE" in CMSIS, "Keil RTX" in "RTOS (API)" and "Startup" in "Device". Then press "OK".



Manage Run-Time Environment

Software Component	Sel.	Variant	Vers...	Description
Board Support				Generic Interfaces for Evaluation and Development Boards
CMSIS				Cortex Microcontroller Software Interface Components
CORE	✓		3.20.0	CMSIS-CORE for Cortex-M, SC000, and SC300
DSP	☐		1.4.1	CMSIS-DSP Library for Cortex-M, SC000, and SC300
RTOS (API)			1.0	CMSIS-RTOS API for Cortex-M, SC000, and SC300
Keil RTX	✓		4.74.0	CMSIS-RTOS RTX implementation for Cortex-M, SC000, and SC300
Device				Startup, System Setup
Startup	✓		1.0.0	System Startup for NXP LPC1700 Series
Drivers				Unified Device Drivers
File System		MDK-Pro	5.0.5	File Access on various storage devices
Graphics		MDK-Pro	5.22.1	User Interface on graphical LCD displays
Network		MDK-Pro	5.0.4	IP Networking using Ethernet or Serial protocols
USB		MDK-Pro	5.0.4	USB Communication with various device classes

Figure 3: Select necessary components to include into our program.

- The project window appears. On the left side of this window, in the "Project" panel, right click on "Source Files" → "Add New Item to Group 'Source Files'".

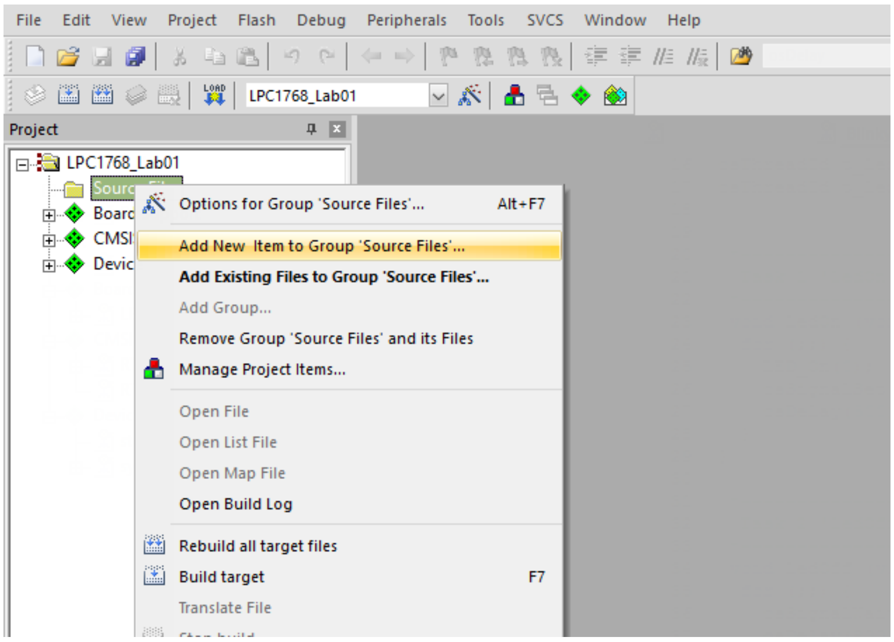
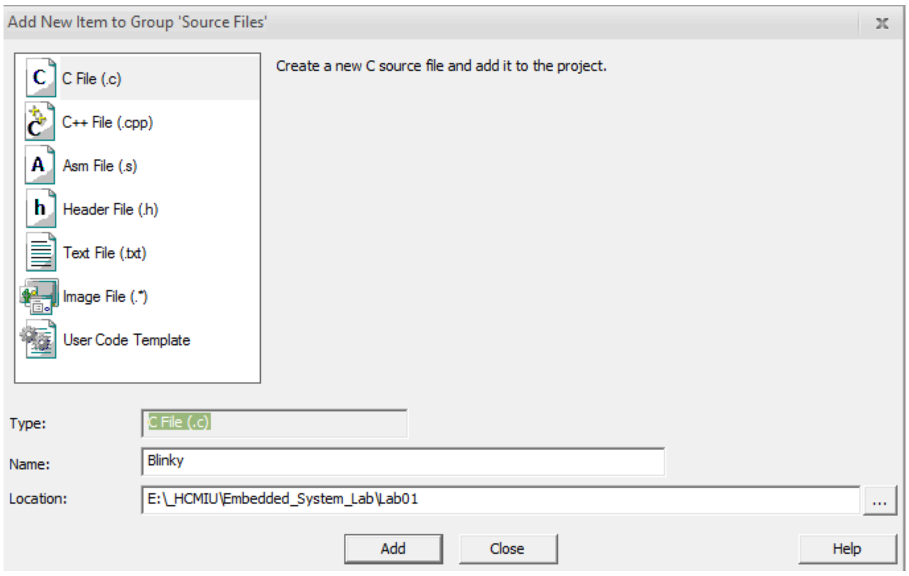

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
LPC1768_Lab01
Project
LPC1768_Lab01
Sourc
Boarc
CMSI
Devic
Options for Group 'Source Files'... Alt+F7
Add New Item to Group 'Source Files'...
Add Existing Files to Group 'Source Files'...
Add Group...
Remove Group 'Source Files' and its Files
Manage Project Items...
Open File
Open List File
Open Map File
Open Build Log
Rebuild all target files
Build target F7
Translate File

Figure 4: Add a new source file to the project.

- Select the type and name of the source file than press “Add”.



Create a new C source file and add it to the project.

Type: C File (.c)

Name: Blinky

Location: E:_HCMIU\Embedded_System_Lab\Lab01

Figure 5: Add a new source file to the project.

1.3.2. Connection

- From the “MCB1700 Quick Start Guild”:

- Connect the white USB cable to the computer (or a USB charger) for power supply.
- Connect the black USB cable to the computer to download and debug embedded programs running on the target hardware.

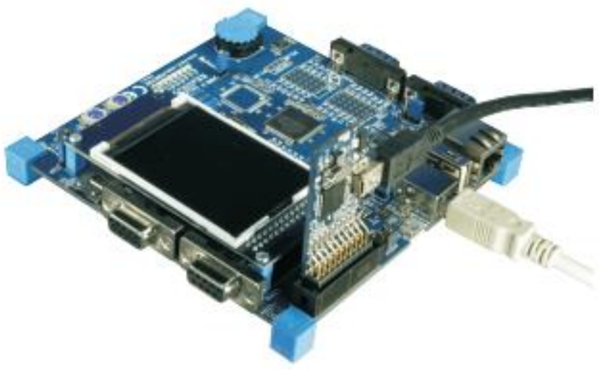


Figure 6: Connection between the board and the computer.

- Simple steps in in setting up peripherals for microcontroller are:

- Setting "Control registers", especially PCONP (Power Control for Peripherals) register.
- Setting "Pin Connect Block registers", especially PINSEL (Pin Function Select) register (from PINSEL0 to PINSEL10).
- Setting corresponding registers to peripherals we are going to use.

In following sections, some common registers will be mentioned in detail.

1.3.3. Build project:

- Build project: To build the project for the target (LPC1768 microcontroller), from the menu of Keil uVision, select *"Project → Build target"*.

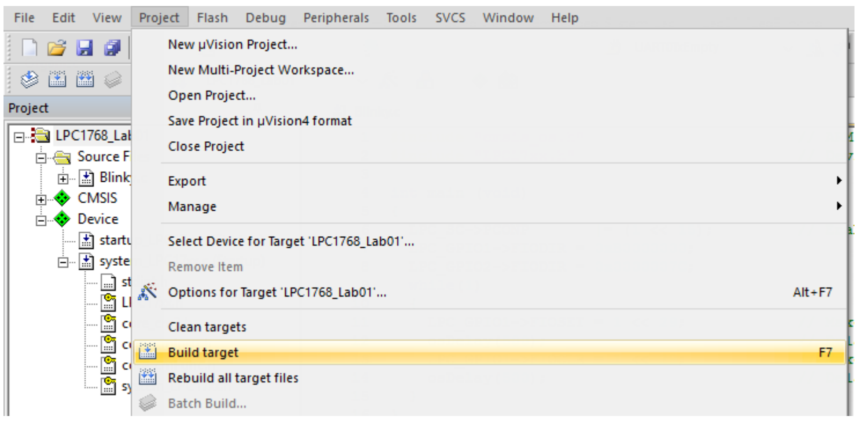


Figure 7: Build project for the target.

- Download to the board: To download the project for the target, from the menu of Keil uVision, select *"Flash → Build Download"*.

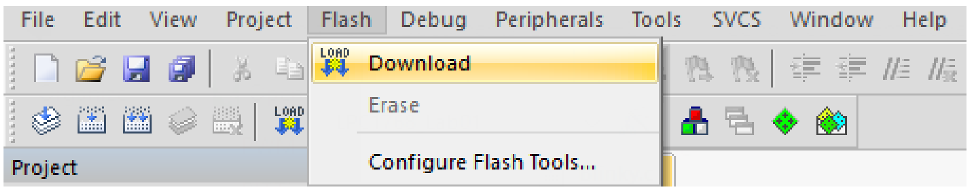
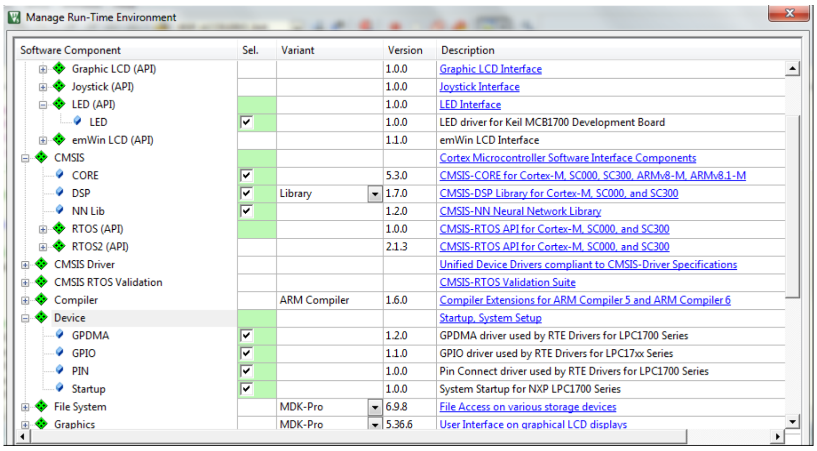


Figure 8: Flash the program into the controller

Chapter II: Projects

2.1. GPIOs and LEDs

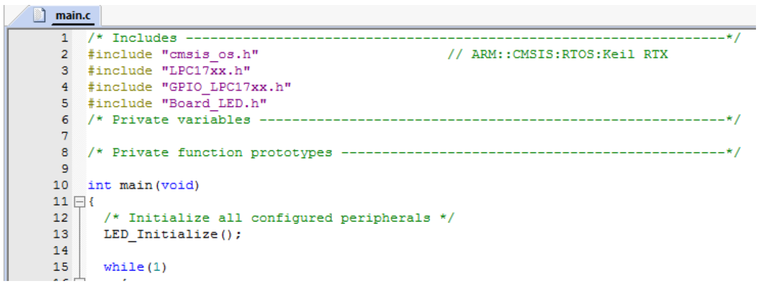
- **Topic: Controls LEDs that are soldered on the board**
- Open “Keil uVision”. Creating new project following above instruction.
- In “Manage Run-Time Environment” window appears, selecting components as follows:



Software Component	Sel.	Variant	Version	Description
Graphic LCD (API)			1.0.0	Graphic LCD Interface
Joystick (API)			1.0.0	Joystick Interface
LED (API)			1.0.0	LED Interface
LED	✓		1.0.0	LED driver for Keil MCB1700 Development Board
emWin LCD (API)			1.1.0	emWin LCD Interface
CMSIS				Cortex Microcontroller Software Interface Components
CORE	✓		5.3.0	CMSIS-CORE for Cortex-M, SC000, SC300, ARMv8-M, ARMv8.1-M
DSP	✓	Library	1.7.0	CMSIS-DSP Library for Cortex-M, SC000, and SC300
NN Lib	✓		1.2.0	CMSIS-NN Neural Network Library
RTOS (API)			1.0.0	CMSIS-RTOS API for Cortex-M, SC000, and SC300
RTOS2 (API)			2.1.3	CMSIS-RTOS API for Cortex-M, SC000, and SC300
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specifications
CMSIS RTOS Validation				CMSIS-RTOS Validation Suite
Compiler		ARM Compiler	1.6.0	Compiler Extensions for ARM Compiler 5 and ARM Compiler 6
Device				Startup, System Setup
GPDMA	✓		1.2.0	GPDMA driver used by RTE Drivers for LPC1700 Series
GPIO	✓		1.1.0	GPIO driver used by RTE Drivers for LPC17xx Series
PIN	✓		1.0.0	Pin Connect driver used by RTE Drivers for LPC1700 Series
Startup	✓		1.0.0	System Startup for NXP LPC1700 Series
File System		MDK-Pro	6.9.8	File Access on various storage devices
Graphics		MDK-Pro	5.36.6	User Interface on graphical LCD displays

Figure 9: Choosing in Ex.1

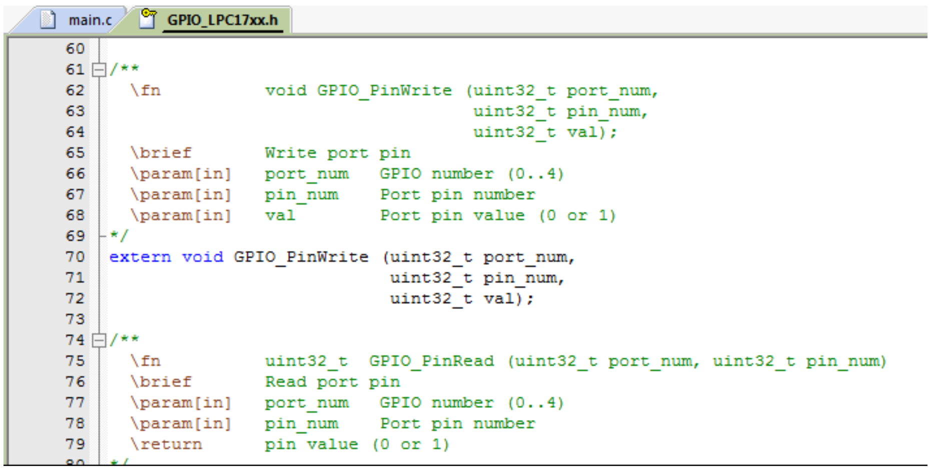
- The project window appears. On the left side of this window, in the “Project” panel, right click on “Source Files” → “Add New Item to Group ‘Source Files’”.
- Declaring libraries and variables as follows:



```c
/* Includes ------------------------------------------------------------------*/
#include "cmsis_os.h"                              // ARM::CMSIS:RTOS:Keil RTX
#include "LPC17xx.h"
#include "GPIO_LPC17xx.h"
#include "Board_LED.h"
/* Private variables ---------------------------------------------------------*/

/* Private function prototypes -----------------------------------------------*/

int main(void)
{
  /* Initialize all configured peripherals */
  LED_Initialize();

  while(1)
```

Figure 10: Opening in Ex.1

- Using the function ***in GPIO_LPC17xx.h:***



```
/**
  \fn          void GPIO_PinWrite (uint32_t port_num,
                                   uint32_t pin_num,
                                   uint32_t val);
  \brief       Write port pin
  \param[in]   port_num   GPIO number (0..4)
  \param[in]   pin_num    Port pin number
  \param[in]   val        Port pin value (0 or 1)
*/
extern void GPIO_PinWrite (uint32_t port_num,
                           uint32_t pin_num,
                           uint32_t val);

/**
  \fn          uint32_t  GPIO_PinRead (uint32_t port_num, uint32_t pin_num)
  \brief       Read port pin
  \param[in]   port_num   GPIO number (0..4)
  \param[in]   pin_num    Port pin number
  \return      pin value (0 or 1)
*/
```

Figure 11: library in Ex.1

- Download to the board and the result:

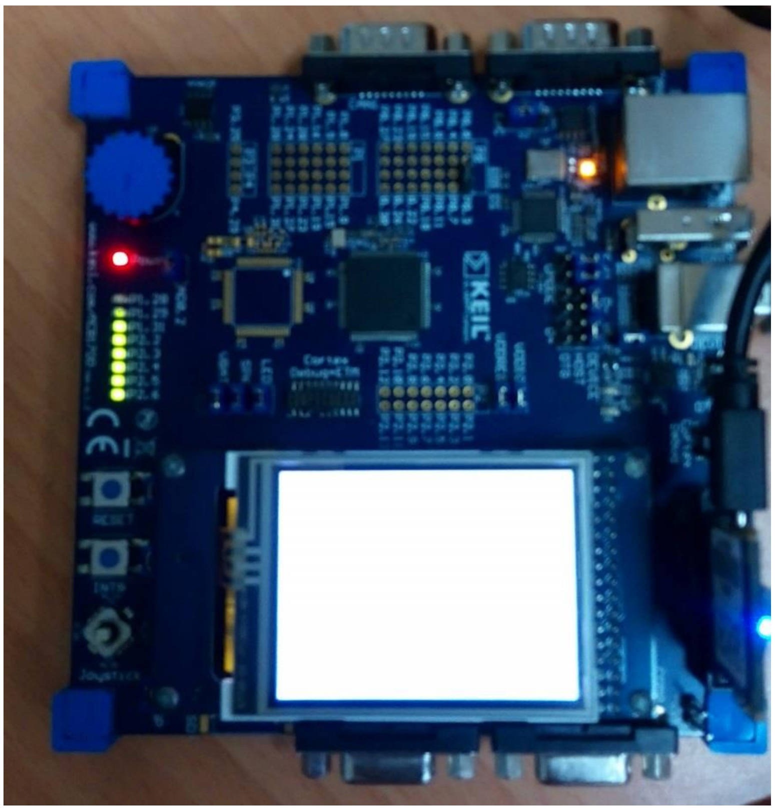
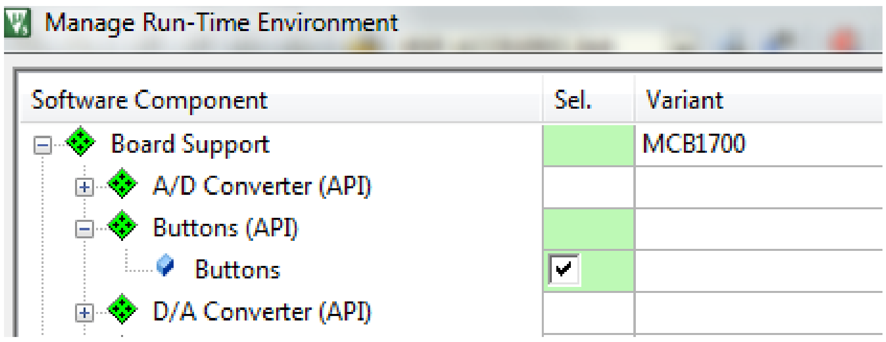


Figure 12: The result of Ex. 1

2.2. Button and LEDs

- **Topic: Using Button to control LEDs**
- Open “Keil uVision”. Creating new project following above instruction.
- In “Manage Run-Time Environment” window appears, selecting components as Example 1 and adding “Button”:

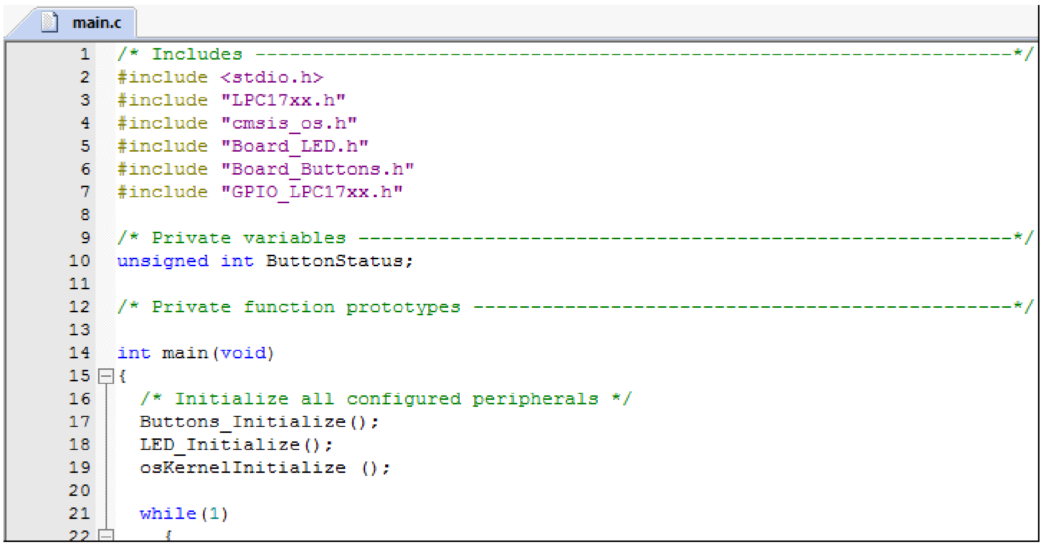


Manage Run-Time Environment

Software Component	Sel.	Variant
Board Support	🟩	MCB1700
A/D Converter (API)		
Buttons (API)	🟩	
Buttons	✓	
D/A Converter (API)		

Figure 13: Choosing in Ex.2

- The project window appears. On the left side of this window, in the “Project” panel, right click on “Source Files” → “Add New Item to Group ‘Source Files’”.
- Declaring libraries and variables as follows:

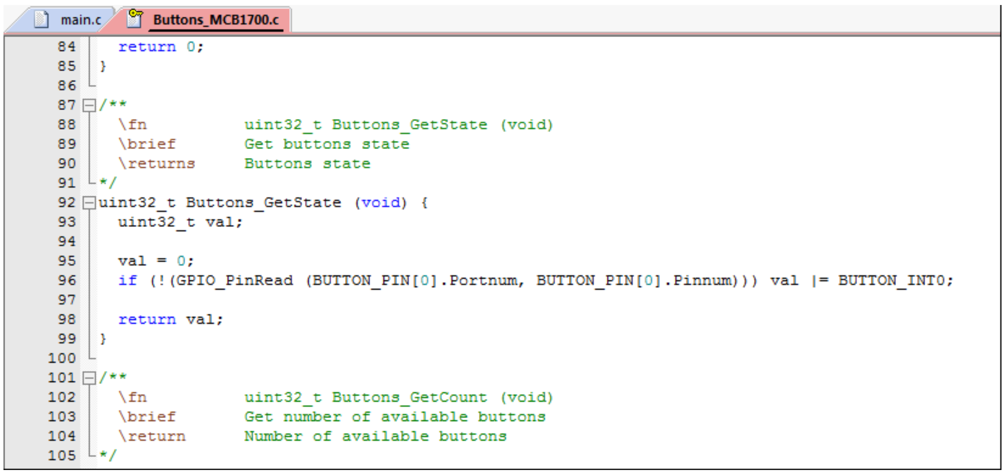


main.c

```
/* Includes ------------------------------------------------------------------*/
#include <stdio.h>
#include "LPC17xx.h"
#include "cmsis_os.h"
#include "Board_LED.h"
#include "Board_Buttons.h"
#include "GPIO_LPC17xx.h"

/* Private variables ---------------------------------------------------------*/
unsigned int ButtonStatus;

/* Private function prototypes -----------------------------------------------*/

int main(void)
{
  /* Initialize all configured peripherals */
  Buttons_Initialize();
  LED_Initialize();
  osKernelInitialize ();

  while(1)
    {
```

Figure 14: Opening in Ex.2

- Using the function *in **Buttons_MBC1700.c:***



```
    return 0;
}

/**
    \fn          uint32_t Buttons_GetState (void)
    \brief       Get buttons state
    \returns     Buttons state
*/
uint32_t Buttons_GetState (void) {
    uint32_t val;

    val = 0;
    if (!(GPIO_PinRead (BUTTON_PIN[0].Portnum, BUTTON_PIN[0].Pinnum))) val |= BUTTON_INT0;

    return val;
}

/**
    \fn          uint32_t Buttons_GetCount (void)
    \brief       Get number of available buttons
    \return      Number of available buttons
*/
```

Figure 15: library in Ex.2

- Download to the board and result:

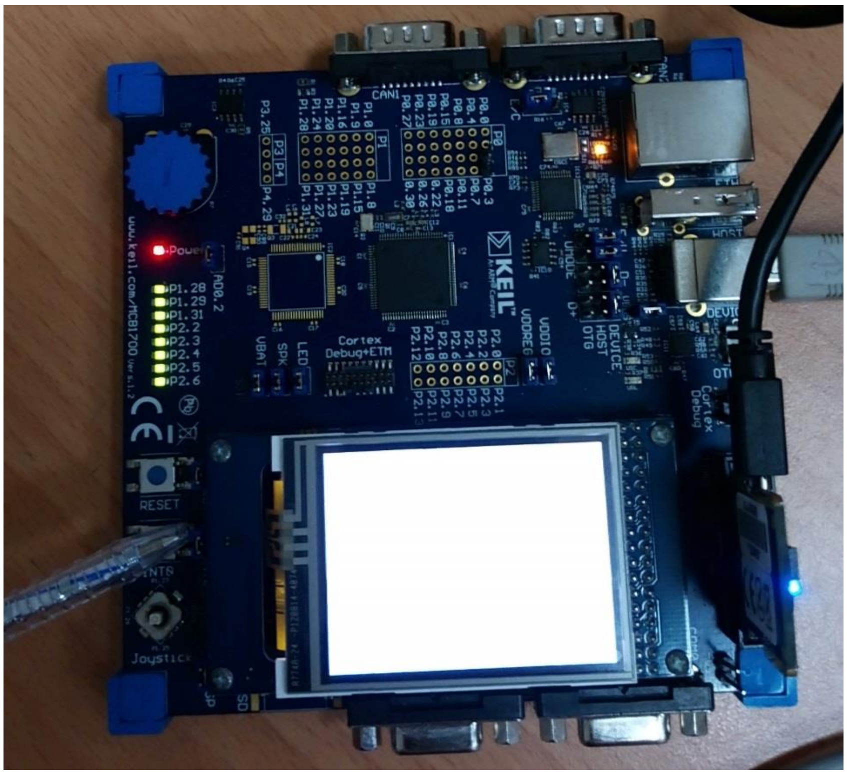
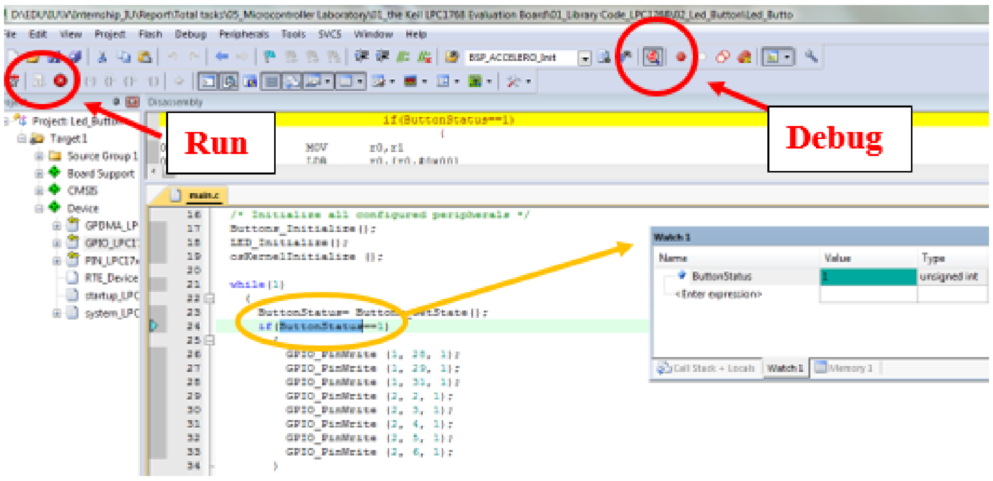


Figure 16: The result of Ex. 2

  - When the button is pressed, LEDs are on
  - When the button is released, LEDs are off

- Using the Debug feature to see the active status. Choosing “Debug” and adding the value on **watch.** Downloading to the board and click **“Run”:**



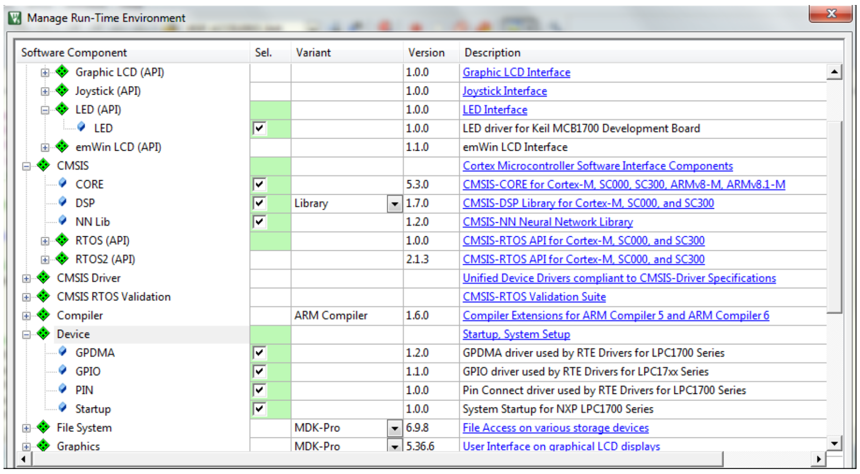
```
if(ButtonStatus==1)
MOV    r0,r1

/* Initialize all configured peripherals */
Buttons_Initialize();
LED_Initialize();
osKernelInitialize ();

while(1)
  {
    ButtonStatus= Buttons_GetState();
    if(ButtonStatus==1)
    {
      GPIO_PinWrite (1, 28, 1);
      GPIO_PinWrite (1, 29, 1);
      GPIO_PinWrite (1, 31, 1);
      GPIO_PinWrite (2, 2, 1);
      GPIO_PinWrite (2, 3, 1);
      GPIO_PinWrite (2, 4, 1);
      GPIO_PinWrite (2, 5, 1);
      GPIO_PinWrite (2, 6, 1);
    }
```

Name	Value	Type
ButtonStatus	1	unsigned int
<Enter expression>		

Figure 17: Debug in Ex.2

2.3. GPIO Library

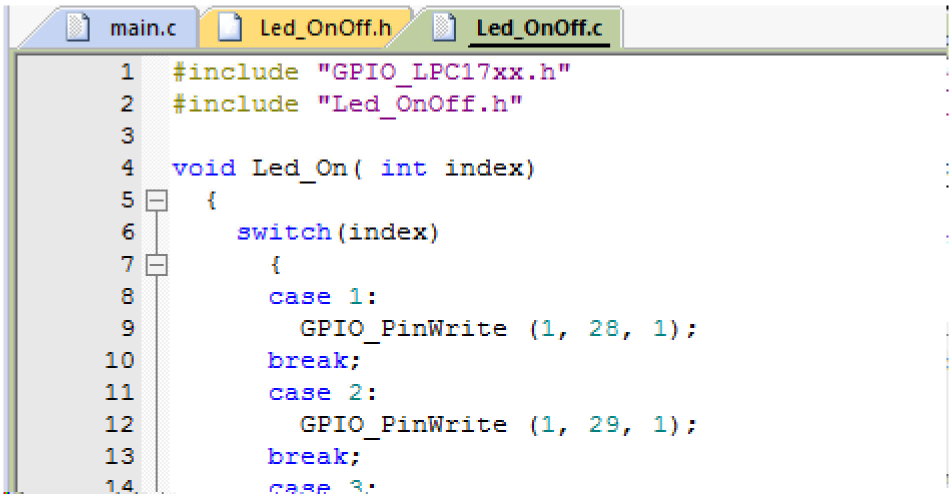
- **Topic: Creating the library to control LEDs**
- Open “Keil uVision”. Creating new project following above instruction.
- In “Manage Run-Time Environment” window appears, selecting components as follows:


Manage Run-Time Environment

Software Component	Sel.	Variant	Version	Description
Graphic LCD (API)			1.0.0	Graphic LCD Interface
Joystick (API)			1.0.0	Joystick Interface
LED (API)			1.0.0	LED Interface
LED	✓		1.0.0	LED driver for Keil MCB1700 Development Board
emWin LCD (API)			1.1.0	emWin LCD Interface
CMSIS				Cortex Microcontroller Software Interface Components
CORE	✓		5.3.0	CMSIS-CORE for Cortex-M, SC000, SC300, ARMv8-M, ARMv8.1-M
DSP	✓	Library	1.7.0	CMSIS-DSP Library for Cortex-M, SC000, and SC300
NN Lib	✓		1.2.0	CMSIS-NN Neural Network Library
RTOS (API)			1.0.0	CMSIS-RTOS API for Cortex-M, SC000, and SC300
RTOS2 (API)			2.1.3	CMSIS-RTOS API for Cortex-M, SC000, and SC300
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specifications
CMSIS RTOS Validation				CMSIS-RTOS Validation Suite
Compiler		ARM Compiler	1.6.0	Compiler Extensions for ARM Compiler 5 and ARM Compiler 6
Device				Startup, System Setup
GPDMA	✓		1.2.0	GPDMA driver used by RTE Drivers for LPC1700 Series
GPIO	✓		1.1.0	GPIO driver used by RTE Drivers for LPC17xx Series
PIN	✓		1.0.0	Pin Connect driver used by RTE Drivers for LPC1700 Series
Startup	✓		1.0.0	System Startup for NXP LPC1700 Series
File System		MDK-Pro	6.9.8	File Access on various storage devices
Graphics		MDK-Pro	5.36.6	User Interface on graphical LCD displays

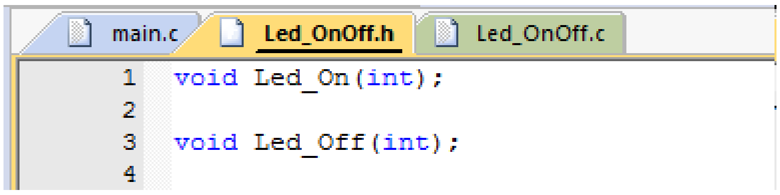
Figure 18: Choosing in Ex.3

- The project window appears. On the left side of this window, in the "Project" panel, right click on "Source Files" → "Add New Item to Group 'Source Files'".
- Creating 2 files as name: ***Led_OnOff.c*** and ***Led_OnOff.h*** :



```c
#include "GPIO_LPC17xx.h"
#include "Led_OnOff.h"

void Led_On( int index)
  {
    switch(index)
      {
      case 1:
        GPIO_PinWrite (1, 28, 1);
      break;
      case 2:
        GPIO_PinWrite (1, 29, 1);
      break;
      case 3:
```

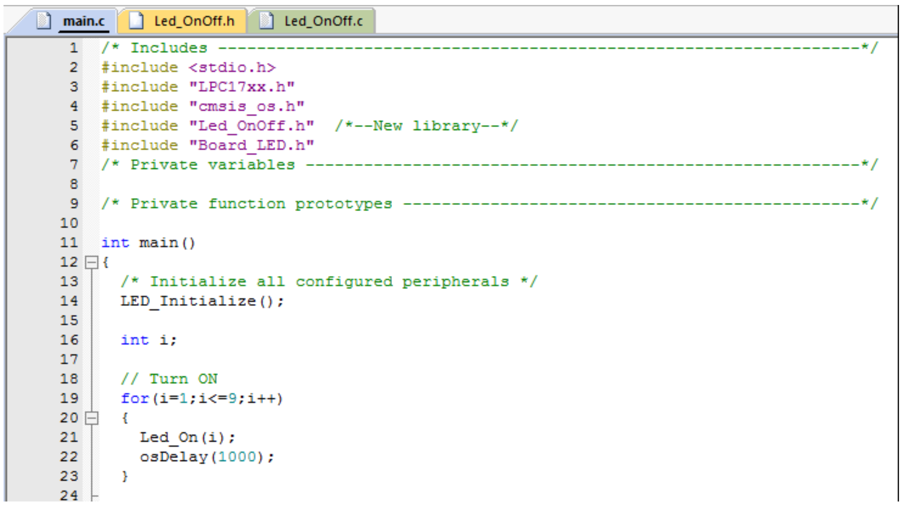
Figure 23: The c file in Ex.3



```c
void Led_On(int);

void Led_Off(int);

```

Figure 19: The h file in Ex.3

- The content as follows:



```c
/* Includes ------------------------------------------------------------------*/
#include <stdio.h>
#include "LPC17xx.h"
#include "cmsis_os.h"
#include "Led_OnOff.h"  /*--New library--*/
#include "Board_LED.h"
/* Private variables ---------------------------------------------------------*/

/* Private function prototypes -----------------------------------------------*/

int main()
{
  /* Initialize all configured peripherals */
  LED_Initialize();

  int i;

  // Turn ON
  for(i=1;i<=9;i++)
  {
    Led_On(i);
    osDelay(1000);
  }

```

Figure 20: The main file in Ex.3

- The result:

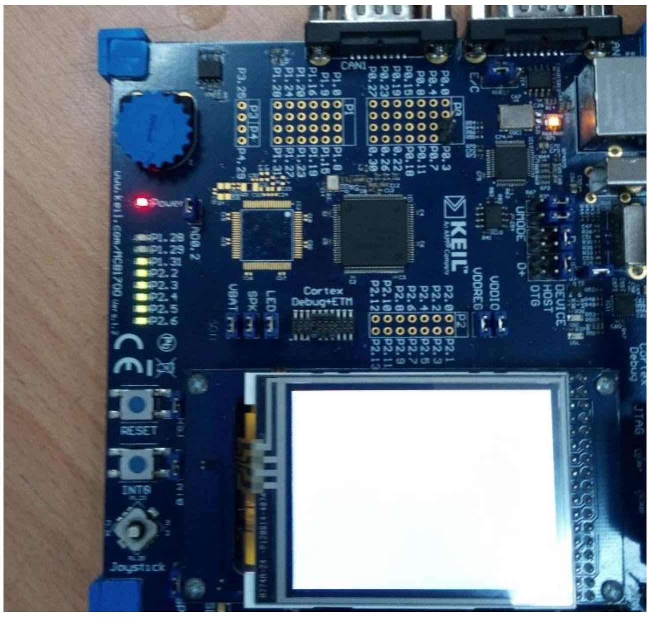
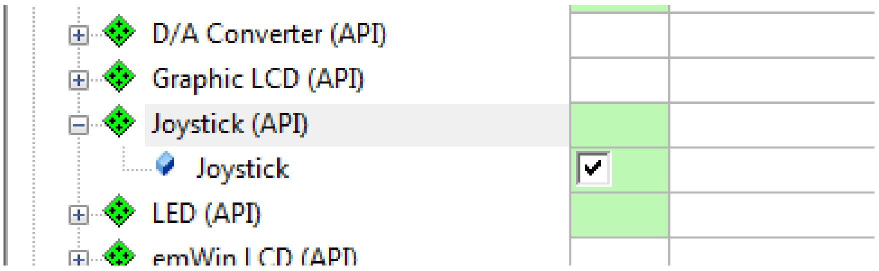


Figure 21: The result in Ex.3

2.4. Joystick

- **Topic: Using Joystick to control LEDs**
- Open "Keil uVision". Creating new project following above instruction.
- In "Manage Run-Time Environment" window appears, selecting components as Example 1 and adding "Joystick":

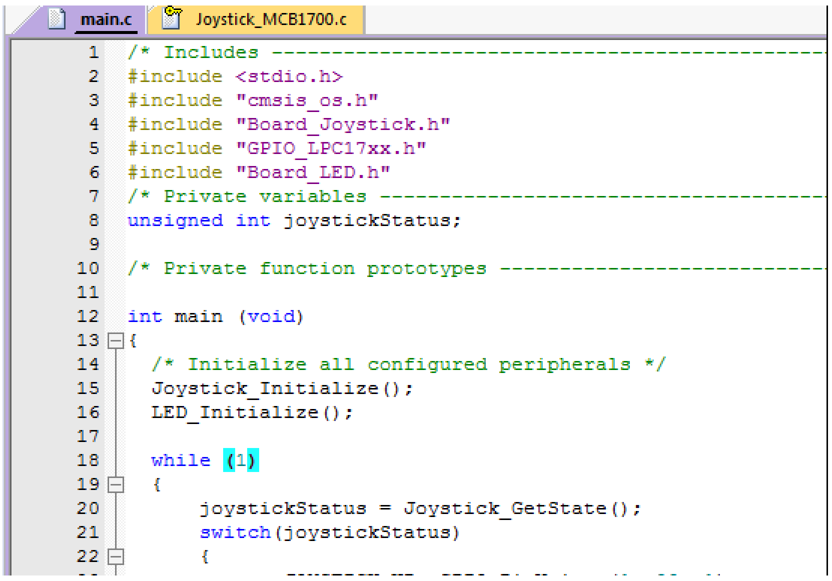


⊞ D/A Converter (API)	
⊞ Graphic LCD (API)	
⊟ Joystick (API)	
Joystick	✓
⊞ LED (API)	
⊞ emWin LCD (API)	

Figure 22: Choosing in Ex.4

- The project window appears. On the left side of this window, in the "Project" panel, right click on "Source Files" → "Add New Item to Group 'Source Files'".

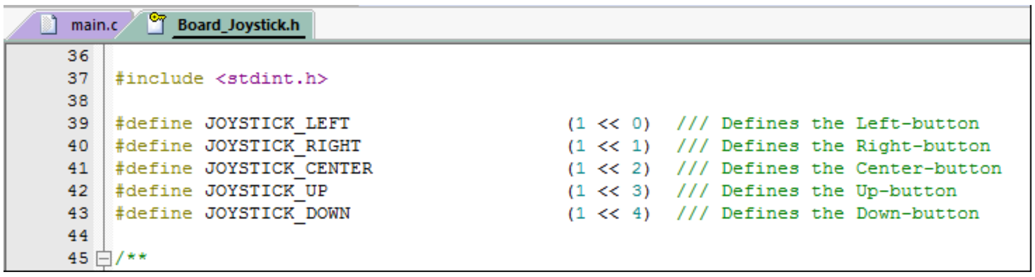
- Declaring libraries and variables as follows:


main.c	Joystick_MCB1700.c

```
/* Includes ----------------------------------------------------
#include <stdio.h>
#include "cmsis_os.h"
#include "Board_Joystick.h"
#include "GPIO_LPC17xx.h"
#include "Board_LED.h"
/* Private variables -------------------------------------------
unsigned int joystickStatus;

/* Private function prototypes ---------------------------------

int main (void)
{
  /* Initialize all configured peripherals */
  Joystick_Initialize();
  LED_Initialize();

  while (1)
  {
      joystickStatus = Joystick_GetState();
      switch(joystickStatus)
      {
```

Figure 23: Opening in Ex.4

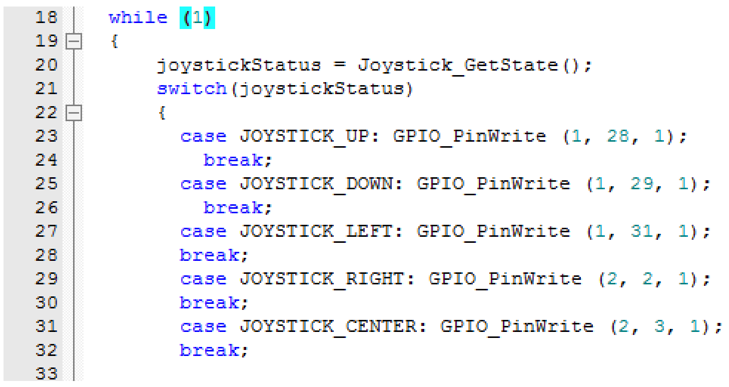
- Using ***Board_Joystick.h:***


main.c	Board_Joystick.h

```
#include <stdint.h>

#define JOYSTICK_LEFT                    (1 << 0)  /// Defines the Left-button
#define JOYSTICK_RIGHT                   (1 << 1)  /// Defines the Right-button
#define JOYSTICK_CENTER                  (1 << 2)  /// Defines the Center-button
#define JOYSTICK_UP                      (1 << 3)  /// Defines the Up-button
#define JOYSTICK_DOWN                    (1 << 4)  /// Defines the Down-button

/**
```

Figure 24: library in Ex.4

- The content in ***main.c*** :



```
  while (1)
  {
      joystickStatus = Joystick_GetState();
      switch(joystickStatus)
      {
        case JOYSTICK_UP: GPIO_PinWrite (1, 28, 1);
          break;
        case JOYSTICK_DOWN: GPIO_PinWrite (1, 29, 1);
          break;
        case JOYSTICK_LEFT: GPIO_PinWrite (1, 31, 1);
        break;
        case JOYSTICK_RIGHT: GPIO_PinWrite (2, 2, 1);
        break;
        case JOYSTICK_CENTER: GPIO_PinWrite (2, 3, 1);
        break;

```

Figure 25: The main file in Ex.4

- Download to the board and result:

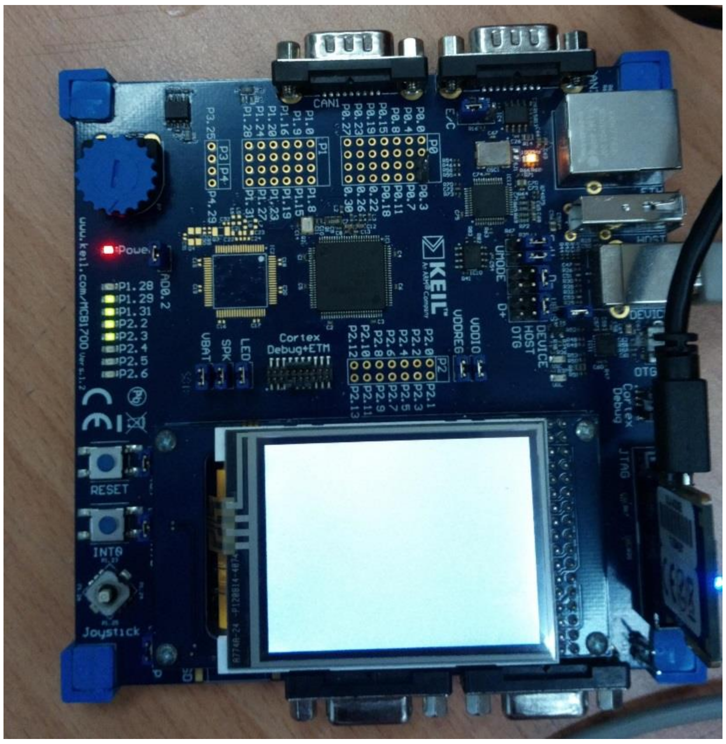


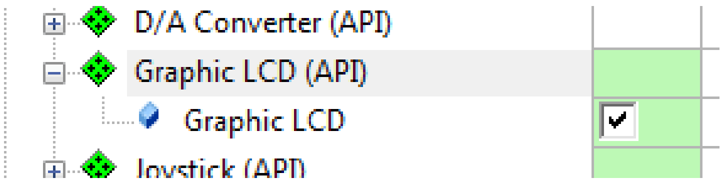
Figure 26: The result of Ex. 4

  - LEDs will turn on according to left, right, up and down control.

- Using the Debug feature to see the active status as example 3

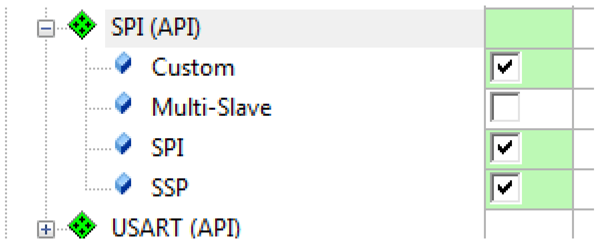
2.5. LCD

- Open “Keil uVision”. Creating new project following above instruction.
- In “Manage Run-Time Environment” window appears, selecting components as follows:



D/A Converter (API)	
Graphic LCD (API)	
Graphic LCD	✓
Joystick (API)	

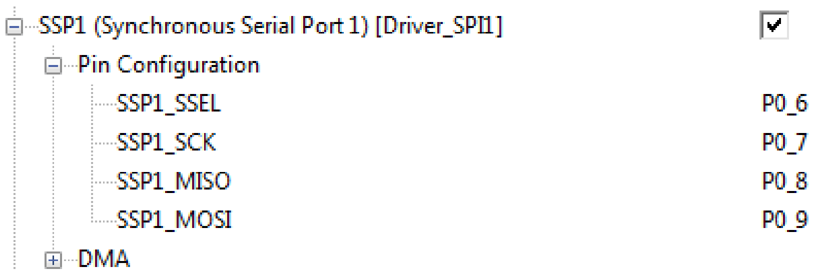
Figure 27. Choosing in Ex. 5



SPI (API)	
Custom	✓
Multi-Slave	☐
SPI	✓
SSP	✓
USART (API)	

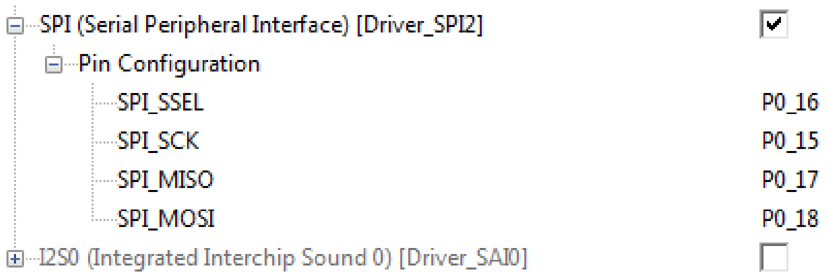
Figure 28. Choosing in Ex. 5

- In ***RTE_Device.h***, setting Pinouts:



SSP1 (Synchronous Serial Port 1) [Driver_SPI1]	✓
Pin Configuration	
SSP1_SSEL	P0_6
SSP1_SCK	P0_7
SSP1_MISO	P0_8
SSP1_MOSI	P0_9
DMA	

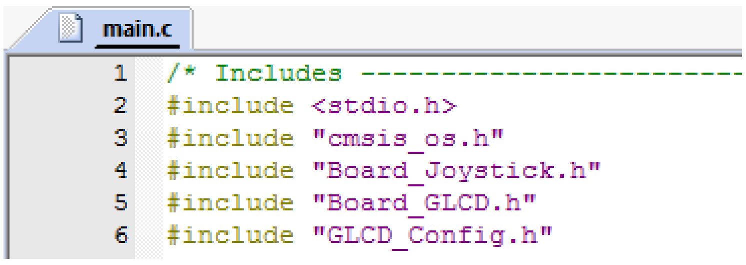
Figure 29. Pinouts



SPI (Serial Peripheral Interface) [Driver_SPI2]	✓
Pin Configuration	
SPI_SSEL	P0_16
SPI_SCK	P0_15
SPI_MISO	P0_17
SPI_MOSI	P0_18
I2S0 (Integrated Interchip Sound 0) [Driver_SAI0]	☐

Figure 30. Pinouts

- The project window appears. On the left side of this window, in the "Project" panel, right click on "Source Files" → "Add New Item to Group 'Source Files'".
- Declaring libraries and variables as follows:

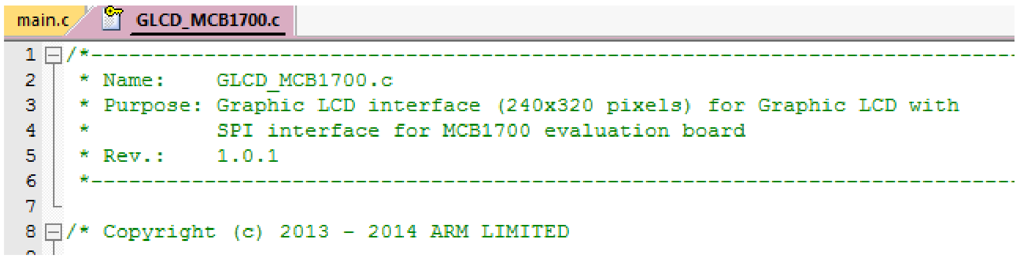


main.c

```
/* Includes ------------------------
#include <stdio.h>
#include "cmsis_os.h"
#include "Board_Joystick.h"
#include "Board_GLCD.h"
#include "GLCD_Config.h"
```

Figure 31. Opening in Ex.5

- Basing **GLCD_MBC1700.c** to write Code



```
main.c    GLCD_MCB1700.c
/*-----------------------------------------------------------------------------
 * Name:    GLCD_MCB1700.c
 * Purpose: Graphic LCD interface (240x320 pixels) for Graphic LCD with
 *          SPI interface for MCB1700 evaluation board
 * Rev.:    1.0.1
 *-----------------------------------------------------------------------------

/* Copyright (c) 2013 - 2014 ARM LIMITED
```

Figure 32. Library for Ex. 5

- Download to the board and result:

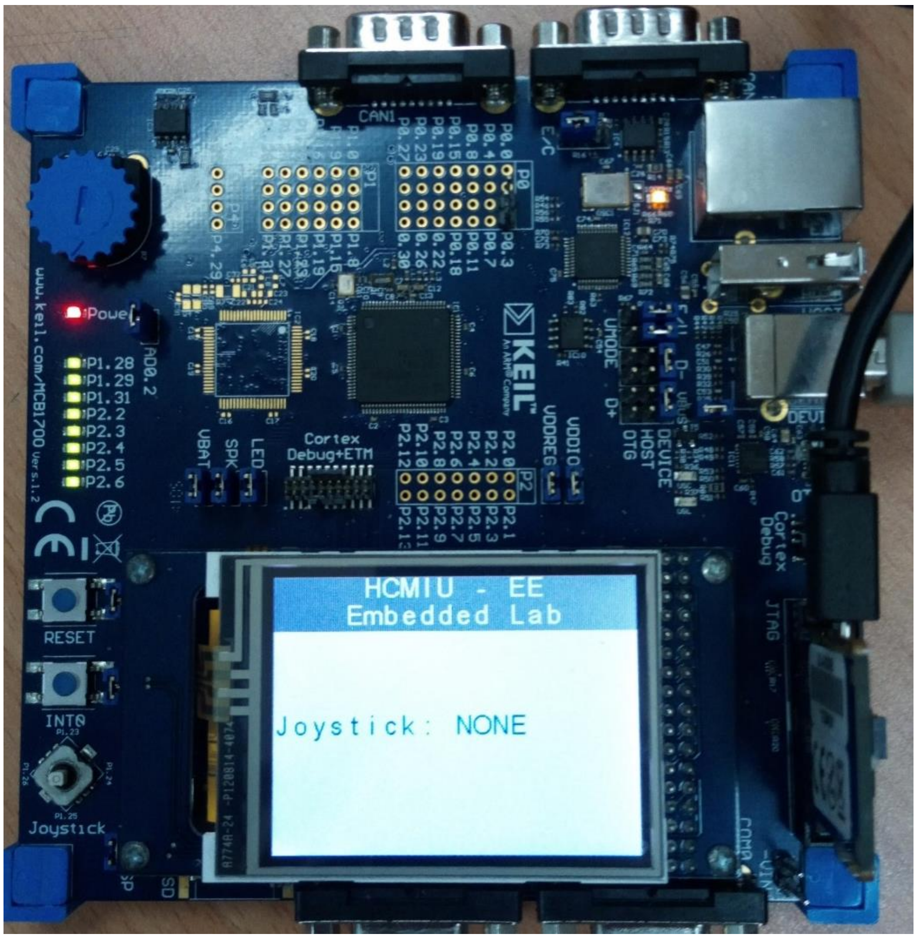
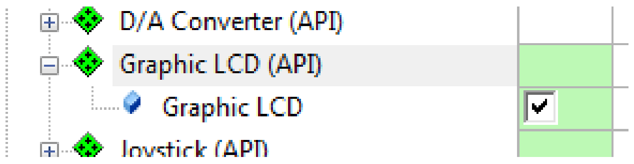


Figure 33. The result of Ex. 5

  - The screen displays the status of the joystick

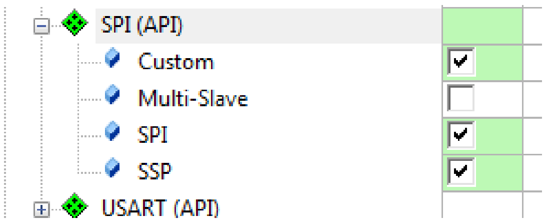
2.6. ADC

- Open “Keil uVision”. Creating new project following above instruction.
- In “Manage Run-Time Environment” window appears, selecting components as follows:



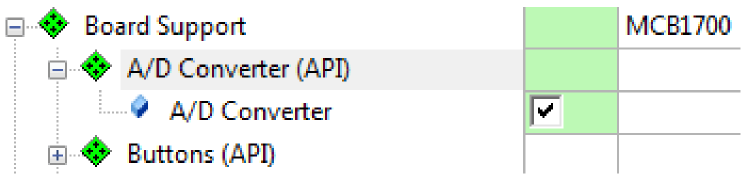
D/A Converter (API)	
Graphic LCD (API)	
Graphic LCD	✓
Joystick (API)	

Figure 34. Choosing in Ex. 6



SPI (API)	
Custom	✓
Multi-Slave	
SPI	✓
SSP	✓
USART (API)	

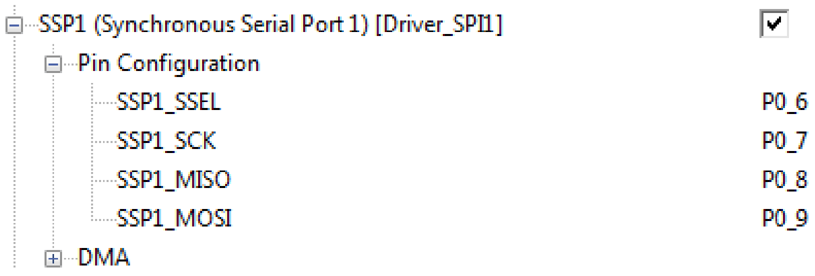
Figure 35. Choosing in Ex. 6



Board Support		MCB1700
A/D Converter (API)		
A/D Converter	✓	
Buttons (API)		

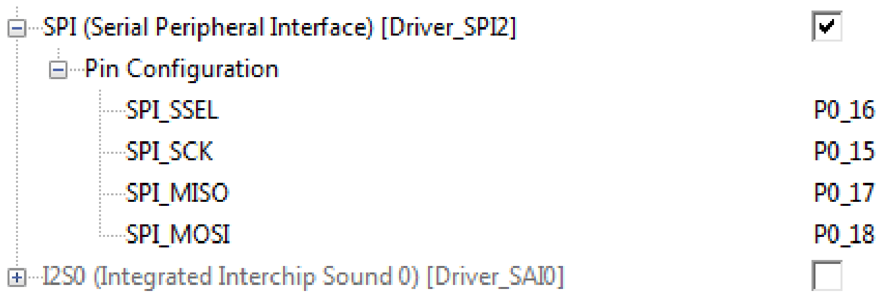
Figure 36. Choosing in Ex. 6

- In ***RTE_Device.h***, setting Pinouts:



SSP1 (Synchronous Serial Port 1) [Driver_SPI1]	✓
Pin Configuration	
SSP1_SSEL	P0_6
SSP1_SCK	P0_7
SSP1_MISO	P0_8
SSP1_MOSI	P0_9
DMA	

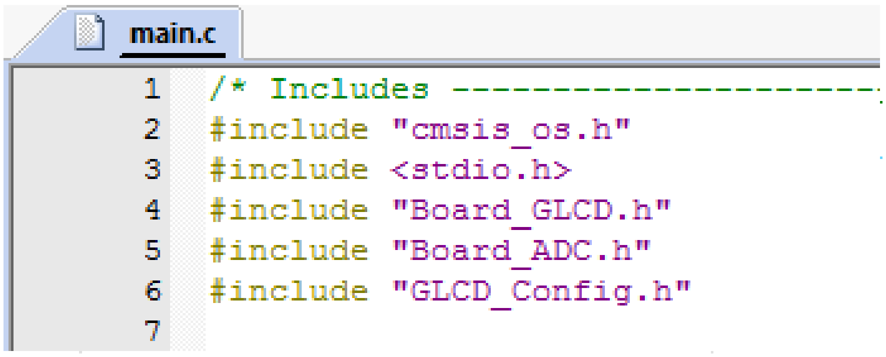
Figure 37. Pinouts



```
SPI (Serial Peripheral Interface) [Driver_SPI2]        ✓
    Pin Configuration
        SPI_SSEL                                       P0_16
        SPI_SCK                                        P0_15
        SPI_MISO                                       P0_17
        SPI_MOSI                                       P0_18
I2S0 (Integrated Interchip Sound 0) [Driver_SAI0]
```

Figure 38. Pinouts

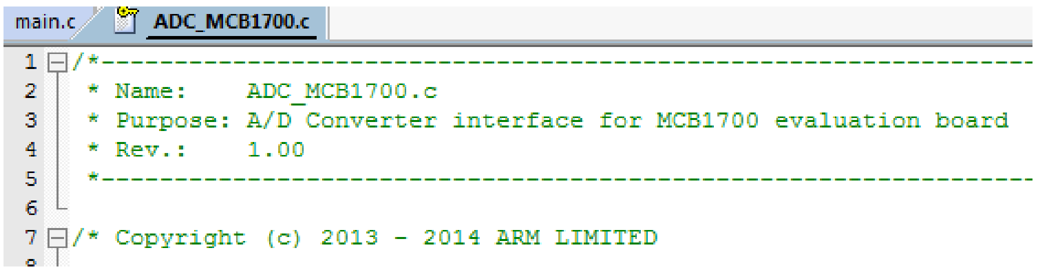
- The project window appears. On the left side of this window, in the "Project" panel, right click on "Source Files" → "Add New Item to Group 'Source Files'".
- Declaring libraries and variables as follows:



```
/* Includes ----------------------
#include "cmsis_os.h"
#include <stdio.h>
#include "Board_GLCD.h"
#include "Board_ADC.h"
#include "GLCD_Config.h"

```

Figure 39. Opening in Ex.6

- Basing ***ADC_MBC1700.c*** to write Code



```
/*-----------------------------------------------------------------
 * Name:    ADC_MCB1700.c
 * Purpose: A/D Converter interface for MCB1700 evaluation board
 * Rev.:    1.00
 *-----------------------------------------------------------------

/* Copyright (c) 2013 - 2014 ARM LIMITED
```

Figure 40. Library for Ex. 6

- Download to the board and result:

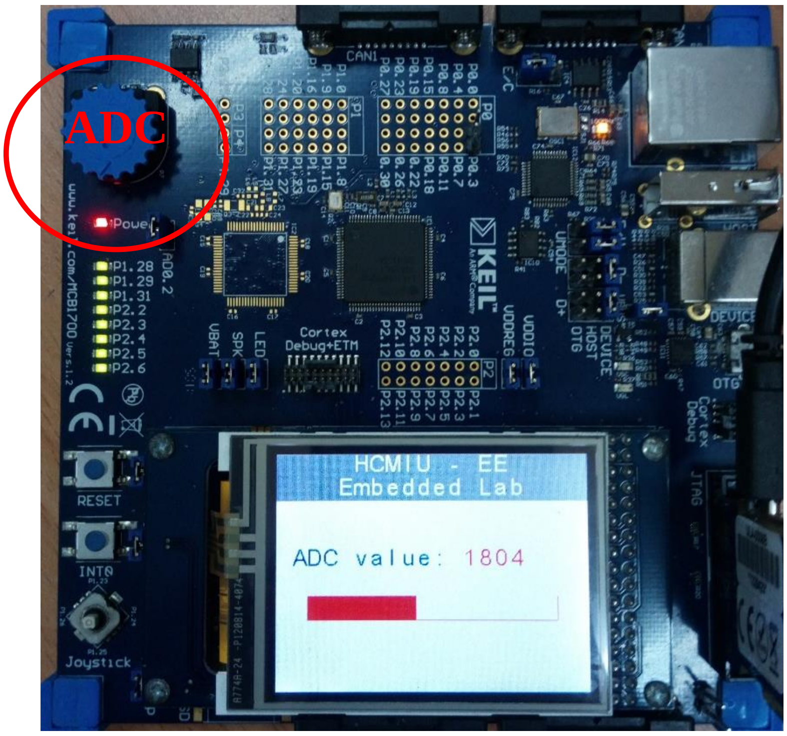
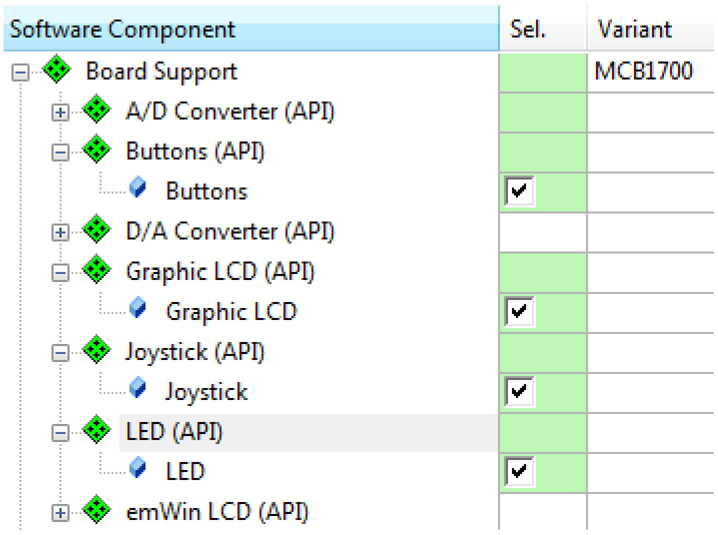


Figure 41: The result of Ex. 6

  - The screen displays the number of ADC
  - Rotating button to adjust

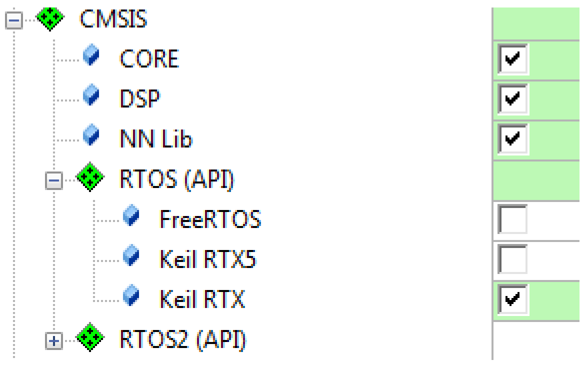
2.7. Control Panel

- Open “Keil uVision”. Creating new project following above instruction.
- In “Manage Run-Time Environment” window appears, selecting components as follows:



Software Component	Sel.	Variant
Board Support		MCB1700
A/D Converter (API)		
Buttons (API)		
Buttons	✓	
D/A Converter (API)		
Graphic LCD (API)		
Graphic LCD	✓	
Joystick (API)		
Joystick	✓	
LED (API)		
LED	✓	
emWin LCD (API)		

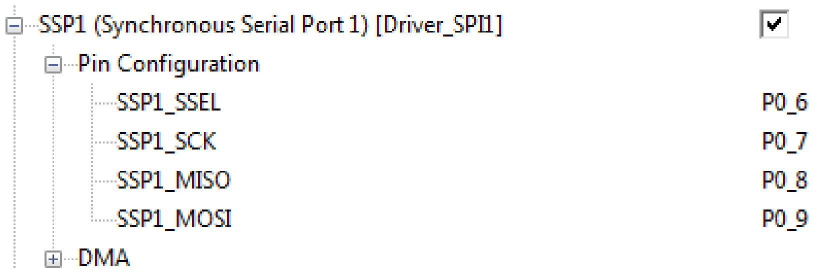
Figure 42. Choosing in Ex. 7



CMSIS	
CORE	✓
DSP	✓
NN Lib	✓
RTOS (API)	
FreeRTOS	☐
Keil RTX5	☐
Keil RTX	✓
RTOS2 (API)	

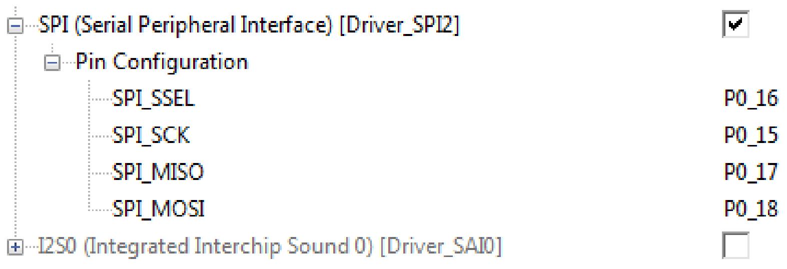
Figure 43. Choosing in Ex. 7

- In ***RTE_Device.h***, setting Pinouts:



SSP1 (Synchronous Serial Port 1) [Driver_SPI1]	✓
Pin Configuration	
SSP1_SSEL	P0_6
SSP1_SCK	P0_7
SSP1_MISO	P0_8
SSP1_MOSI	P0_9
DMA	

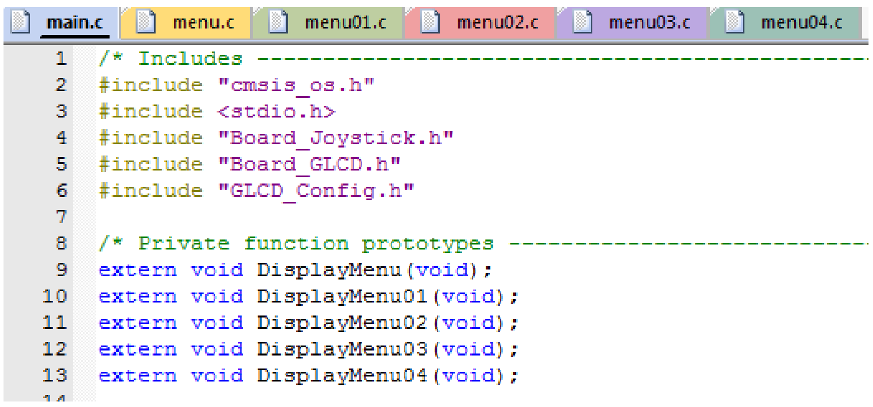
Figure 44. Pinouts



```
SPI (Serial Peripheral Interface) [Driver_SPI2]        ✓
  Pin Configuration
    SPI_SSEL                                           P0_16
    SPI_SCK                                            P0_15
    SPI_MISO                                           P0_17
    SPI_MOSI                                           P0_18
I2S0 (Integrated Interchip Sound 0) [Driver_SAI0]      ☐
```

Figure 45. Pinouts

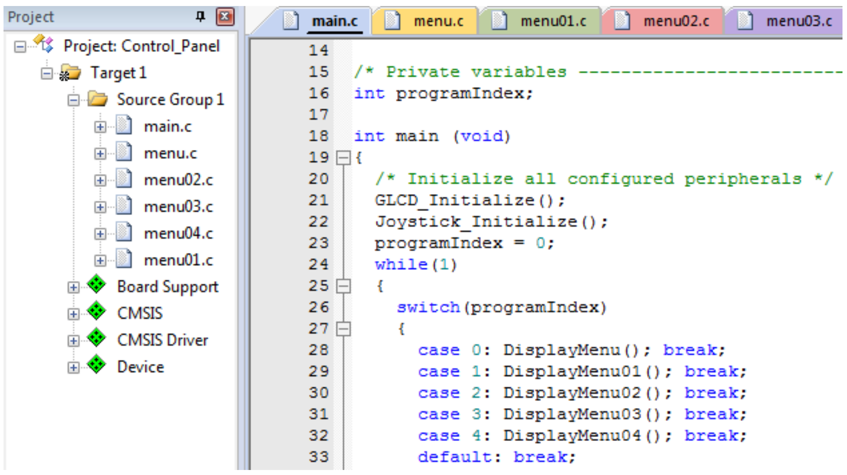
- The project window appears. On the left side of this window, in the “Project” panel, right click on “Source Files” → “Add New Item to Group ‘Source Files’”.
- Declaring libraries and variables as follows:



```c
/* Includes ----------------------------------------------------
#include "cmsis_os.h"
#include <stdio.h>
#include "Board_Joystick.h"
#include "Board_GLCD.h"
#include "GLCD_Config.h"

/* Private function prototypes ---------------------------
extern void DisplayMenu(void);
extern void DisplayMenu01(void);
extern void DisplayMenu02(void);
extern void DisplayMenu03(void);
extern void DisplayMenu04(void);
```

Figure 46. Opening in Ex.7

- The structure of project:



```c

/* Private variables ------------------------
int programIndex;

int main (void)
{
  /* Initialize all configured peripherals */
  GLCD_Initialize();
  Joystick_Initialize();
  programIndex = 0;
  while(1)
  {
    switch(programIndex)
    {
      case 0: DisplayMenu(); break;
      case 1: DisplayMenu01(); break;
      case 2: DisplayMenu02(); break;
      case 3: DisplayMenu03(); break;
      case 4: DisplayMenu04(); break;
      default: break;
```

Figure 47. The structure for Ex. 7

- Download to the board and result:

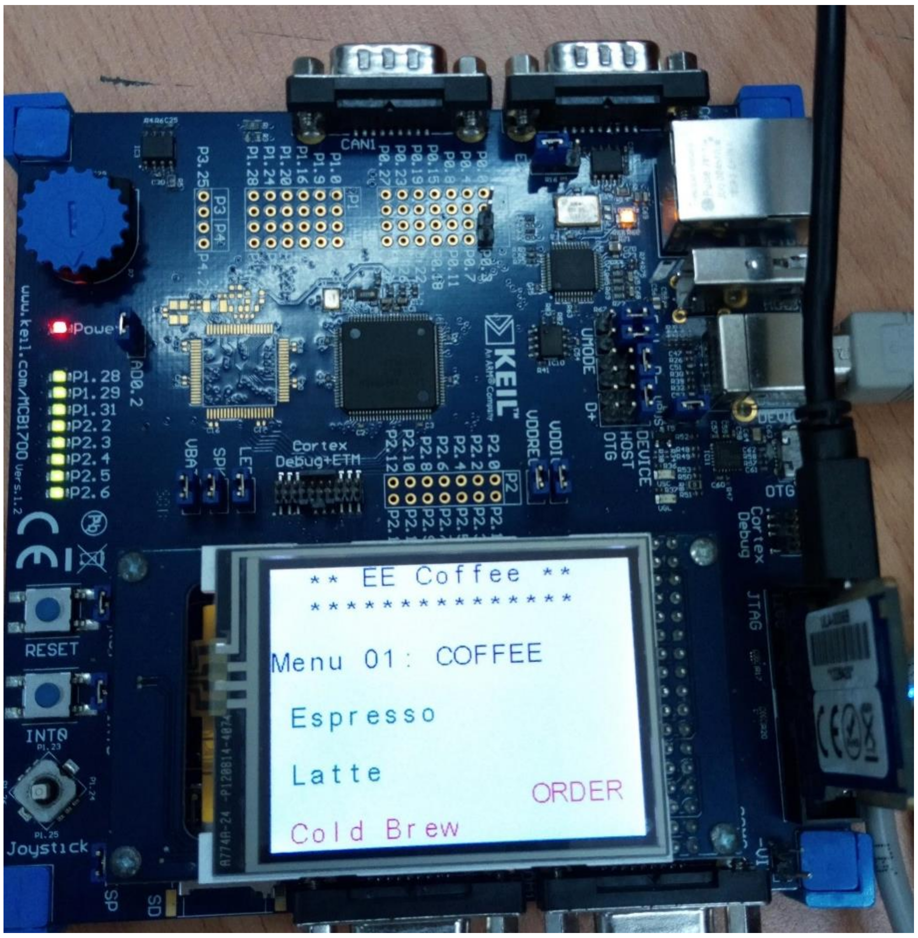


Figure 48: The result of Ex. 7

  - The screen displays the menu
  - Press down Joystick to order