|  |
| --- |
| **EMBEDDED SYSTEM LABORATORY** |
| **LAB 3** |

**MULTITASK PROGRAMMING FOR ARM MICROCONTROLLER WITH ADC INTERFACE**

Nguyễn Gia Cát Tường ITITIU21117

### I. LAB OBJECTIVES

### - In this Lab students will learn about ARM-CORTEX M3 (LPC1768) Microcontroller.

### - This Lab experiments are intended to implement basic ADC of ARM-CORTEX M3 Microcotroller to pheriperal devices in MB1700 Kit and write C code programming to control these devices.

### II. PRE-LAB : ADC Register Review

A screenshot of a computer program

Description automatically generated

A screenshot of a computer registration form

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

A black and white text on a white background

Description automatically generated

### III. LAB PROCERUCE

### The LPC1768 Microconttroler KIT using 100MHz system clock.

### III.1 Lab Experiment 1 : Write the C code to use GLCD Driver Library in GLCD to draw a rectangle 100x100 pixel at position x=50, y=50, then move the rectangle in different position of the LCD screen with moving delay 1 second using timer 1 polling method; (done lab3-ex)

### #include <LPC17xx.h> #include <PIN\_LPC17xx.h>

### #include <GPIO\_LPC17xx.h>

### #include <GLCD\_Config.h>

### #include <Board\_GLCD.h>

### #include <GLCD\_Fonts.h>

### #include <Driver\_SPI.h>

### #define PRESCALE (25000-1) // PCLK=50.000.000/50.000 = 1000Hz=> T\_PCLK=1ms

### void delayms(unsigned int milliseconds);

### void Timer1\_Init(void);

### int x = 50;

### int y = 50;

### int main( void ) {

### Timer1\_Init();

### GLCD\_Initialize();

### GLCD\_SetBackgroundColor (GLCD\_COLOR\_GREEN);

### GLCD\_ClearScreen();

### 

### while ( 1 ) {

### delayms(1000);

### x = x+20;

### GLCD\_SetForegroundColor(GLCD\_COLOR\_BLACK);

### GLCD\_DrawRectangle(x,y,50,100);

### delayms(500);

### GLCD\_SetForegroundColor(GLCD\_COLOR\_GREEN);

### GLCD\_DrawRectangle(x,y,50,100);

### if(x == 100){

### y += 20;

### }

### }

### }

### void Timer1\_Init(void)

### {

### LPC\_SC->PCONP |= (1<<2); // turn on Timer1 module

### LPC\_SC->PCLKSEL0 &=~(0x03<<5);//0b010000; systemcore clock/4= 100MHz/4 =25MHz

### LPC\_TIM1->PR = PRESCALE; // 25 000 000/25 000 = 1000Hz => t = 1/1000 = 1ms

### LPC\_TIM1->TCR = 0x02;

### }

### 

### void delayms(unsigned int milliseconds)

### {

### LPC\_TIM1->TCR = 0x02; // reset counter b1 of TCR =1

### LPC\_TIM1->TCR = 0x01; // enable couter to count

### while(LPC\_TIM1->TC < milliseconds);

### LPC\_TIM1->TCR = 0x00;

### }

### 

### 

### III.2 Lab Experiment 2 : Write the C code to use GLCD Driver Library in GLCD to draw a bar graph 100x100 pixel at position x=150, y=5; then move the bar graph in different position of the LCD screen with joystick controlled by four button P1.23 P1.24 P1.25 P1.26 (done)

#include <LPC17xx.h>

#include <PIN\_LPC17xx.h>

#include <GPIO\_LPC17xx.h>

#include <GLCD\_Config.h>

#include <Board\_GLCD.h>

#include <GLCD\_Fonts.h>

#define PRESCALE (50000-1) // Define the prescale value for Timer0

void delayms(unsigned int milliseconds); // Function prototype for delay in milliseconds

void Timer0\_Init(void); // Function prototype for Timer0 initialization

#define Joystick\_key\_left 23

#define Joystick\_key\_up 24

#define Joystick\_key\_right 25

#define Joystick\_key\_down 26

int x = 1;

int y = 1;

int main(void) {

LPC\_GPIO1->FIODIR &= ~((1 << Joystick\_key\_up) | (1 << Joystick\_key\_down) | (1 << Joystick\_key\_left) | (1 << Joystick\_key\_right));

// Set joystick pins as inputs

Timer0\_Init();

GLCD\_Initialize();

GLCD\_ClearScreen();

GLCD\_SetFont(&GLCD\_Font\_16x24); // Set the GLCD font to 16x24

GLCD\_SetBackgroundColor(GLCD\_COLOR\_BLACK); // Set the background color to black

GLCD\_SetBackgroundColor(GLCD\_COLOR\_WHITE); // Set the foreground color to white

GLCD\_DrawRectangle(x, y, 100, 100); // Draw a rectangle at (x, y) with width and height of 100

while (1) { // Infinite loop

if ((LPC\_GPIO1->FIOPIN & (1 << Joystick\_key\_up)) == 0) { // Check if the up joystick key is pressed

delayms(100); // Delay for debouncing

GLCD\_SetForegroundColor(GLCD\_COLOR\_BLACK); // Set the foreground color to black

GLCD\_DrawRectangle(x, y, 100, 100); // Draw the rectangle in black to "erase" it

delayms(100); // Delay for smooth drawing

GLCD\_SetForegroundColor(GLCD\_COLOR\_WHITE); // Set the foreground color to white

GLCD\_DrawRectangle(x, y, 100, 100); // Draw the rectangle in white

x += 5; // Move the rectangle up by increasing x coordinate

GLCD\_SetForegroundColor(GLCD\_COLOR\_BLACK); // Set the foreground color to black

GLCD\_DrawRectangle(x, y, 100, 100); // Draw the rectangle at the new position

}

if ((LPC\_GPIO1->FIOPIN & (1 << Joystick\_key\_down)) == 0) { // Check if the down joystick key is pressed

delayms(100); // Delay for debouncing

GLCD\_SetForegroundColor(GLCD\_COLOR\_BLACK); // Set the foreground color to black

GLCD\_DrawRectangle(x, y, 100, 100); // Draw the rectangle in black to "erase" it

delayms(100); // Delay for smooth drawing

GLCD\_SetForegroundColor(GLCD\_COLOR\_WHITE); // Set the foreground color to white

GLCD\_DrawRectangle(x, y, 100, 100); // Draw the rectangle in white

x -= 5; // Move the rectangle down by decreasing x coordinate

GLCD\_SetForegroundColor(GLCD\_COLOR\_BLACK); // Set the foreground color to black

GLCD\_DrawRectangle(x, y, 100, 100); // Draw the rectangle at the new position

}

if ((LPC\_GPIO1->FIOPIN & (1 << Joystick\_key\_left)) == 0) { // Check if the left joystick key is pressed

delayms(100); // Delay for debouncing

GLCD\_SetForegroundColor(GLCD\_COLOR\_BLACK); // Set the foreground color to black

GLCD\_DrawRectangle(x, y, 100, 100); // Draw the rectangle in black to "erase" it

delayms(100); // Delay for smooth drawing

GLCD\_SetForegroundColor(GLCD\_COLOR\_WHITE); // Set the foreground color to white

GLCD\_DrawRectangle(x, y, 100, 100); // Draw the rectangle in white

y -= 5; // Move the rectangle left by decreasing y coordinate

GLCD\_SetForegroundColor(GLCD\_COLOR\_BLACK); // Set the foreground color to black

GLCD\_DrawRectangle(x, y, 100, 100); // Draw the rectangle at the new position

}

if ((LPC\_GPIO1->FIOPIN & (1 << Joystick\_key\_right)) == 0) { // Check if the right joystick key is pressed

delayms(100); // Delay for debouncing

GLCD\_SetForegroundColor(GLCD\_COLOR\_BLACK); // Set the foreground color to black

GLCD\_DrawRectangle(x, y, 100, 100); // Draw the rectangle in black to "erase" it

delayms(100); // Delay for smooth drawing

GLCD\_SetForegroundColor(GLCD\_COLOR\_WHITE); // Set the foreground color to white

GLCD\_DrawRectangle(x, y, 100, 100); // Draw the rectangle in white

y += 5; // Move the rectangle right by increasing y coordinate

GLCD\_SetForegroundColor(GLCD\_COLOR\_BLACK); // Set the foreground color to black

GLCD\_DrawRectangle(x, y, 100, 100); // Draw the rectangle at the new position

}

}

}

void Timer0\_Init(void) {

LPC\_SC->PCONP |= (1 << 1); // Turn on the power for Timer0

LPC\_SC->PCLKSEL0 |= (1 << 2); // Set PCLKSEL0[3:2] to "10" for system clock/2 (100MHz/2 = 50MHz)

LPC\_TIM0->CTCR = 0x0; // Set Timer0 to Timer Mode

LPC\_TIM0->PR = PRESCALE; // Set the prescaler to 50000-1

LPC\_TIM0->TCR = 0x02; // Reset Timer

}

void delayms(unsigned int milliseconds) {

LPC\_TIM0->TCR = 0x02; // Reset Timer

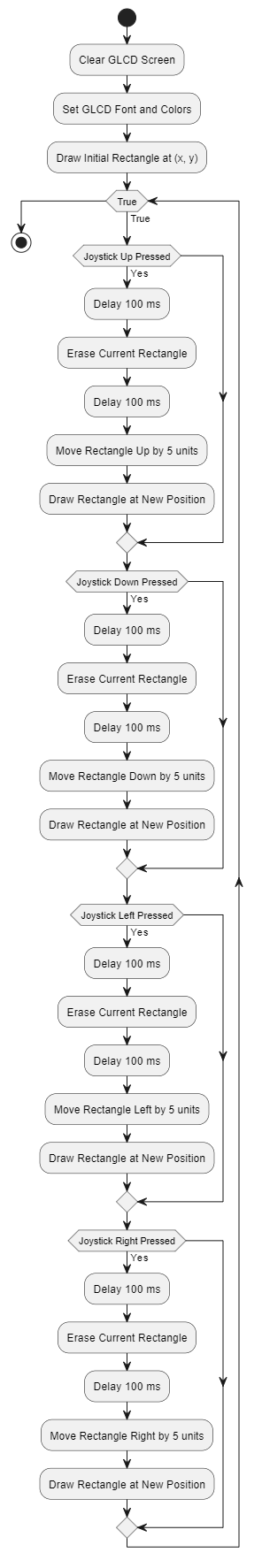
LPC\_TIM0->TCR = 0x01; // Enable Timer

while (LPC\_TIM0->TC < milliseconds); // Wait until the timer counter reaches the specified milliseconds

LPC\_TIM0->TCR = 0x00; // Disable Timer

}

**FLOW CHART**



### III.3 Lab Experiment 3 : Write the C code to use GLCD Driver Library in GLCD to display your name at position x=10, y=160 ; then scroll up your name in the screen. (done)

### #include <LPC17xx.h

### #include <PIN\_LPC17xx.h>

### #include <GPIO\_LPC17xx.h>

### #include <GLCD\_Config.h>

### #include <Board\_GLCD.h>

### #include <GLCD\_Fonts.h>

### #include <Driver\_SPI.h>

### char myString[] = {"HEHE\_BOIZ"}; // Define the text string to be displayed

### int y = 0; // Initialize y-coordinate for text position

### int main(void)

### {

### GLCD\_Initialize(); // Initialize the GLCD display

### // Set font and background colors

### GLCD\_SetFont(&GLCD\_Font\_16x24); // Set the font to 16x24 pixels

### GLCD\_SetBackgroundColor(GLCD\_COLOR\_LIGHT\_GREY); // Set background to light grey (unused)

### GLCD\_SetBackgroundColor(GLCD\_COLOR\_WHITE); // Set background to white

### for (int i = 0; i < 160; i++)

### { // Loop to create scrolling effect

### y = y - 5; // Move text upwards by 5 pixels

### GLCD\_DrawString(10, y, myString); // Draw the string at new position (10, y)

### }

### return 0; // Added return statement

### }

### III.5 Lab Experiment 5 : Write the C code to get data From ADC channel 2 using polling method then display 8-bit high value result to 8 bit LED in the Experiment KIT . (done)

#include <LPC17xx.h>

#include <PIN\_LPC17xx.h>

#include <GPIO\_LPC17xx.h>

#include <GLCD\_Config.h>

#include <Board\_GLCD.h>

#include <GLCD\_Fonts.h>

#include <Driver\_SPI.h>

#include <stdio.h>

char myString[]={"hello, world"};

char ADC\_String[20]=" ";

#define VREF 3.3 //Reference Voltage at VREFP pin, given VREFN = 0V(GND)

#define ADC\_CLK\_EN (1<<12)

#define SEL\_AD0\_2 (1<<6) //Select Channel AD0.0

#define CLKDIV 1 //ADC clock-divider (ADC\_CLOCK=PCLK/CLKDIV+1) = 12.5Mhz

//25Mhz PCLK

#define PWRUP (1<<21) //setting it to 0 will power it down

#define START\_CNV (1<<24) //001 for starting the conversion immediately

#define ADC\_DONE (1U<<31) //define it as unsigned value Done Bit

#define ADCR\_SETUP\_SCM ((CLKDIV<<8) | PWRUP)

int main( void ) {

GLCD\_Initialize() ;

GLCD\_SetFont(&GLCD\_Font\_16x24) ;

GLCD\_SetBackgroundColor (GLCD\_COLOR\_GREEN) ;

GLCD\_SetBackgroundColor (GLCD\_COLOR\_LIGHT\_GREY);

GLCD\_ClearScreen() ;

LPC\_SC->PCONP |= ADC\_CLK\_EN; //Enable ADC clock

LPC\_ADC->ADCR = ADCR\_SETUP\_SCM | SEL\_AD0\_2;

LPC\_PINCON->PINSEL0 |= (1<<7) ; //select AD0.2 for P0.25

int result = 0;

int bar\_value;

float volts = 0;

while ( 1 ) {

LPC\_ADC->ADCR |= START\_CNV; //Start new Conversion

while((LPC\_ADC->ADDR6 & ADC\_DONE) == 0); //Wait untill conversion is

//finished

result = (LPC\_ADC->ADDR6>>4) & 0xFFF; //12 bit Mask to extract

//result

volts = (result\*VREF)/4096.0; //Convert result to Voltage

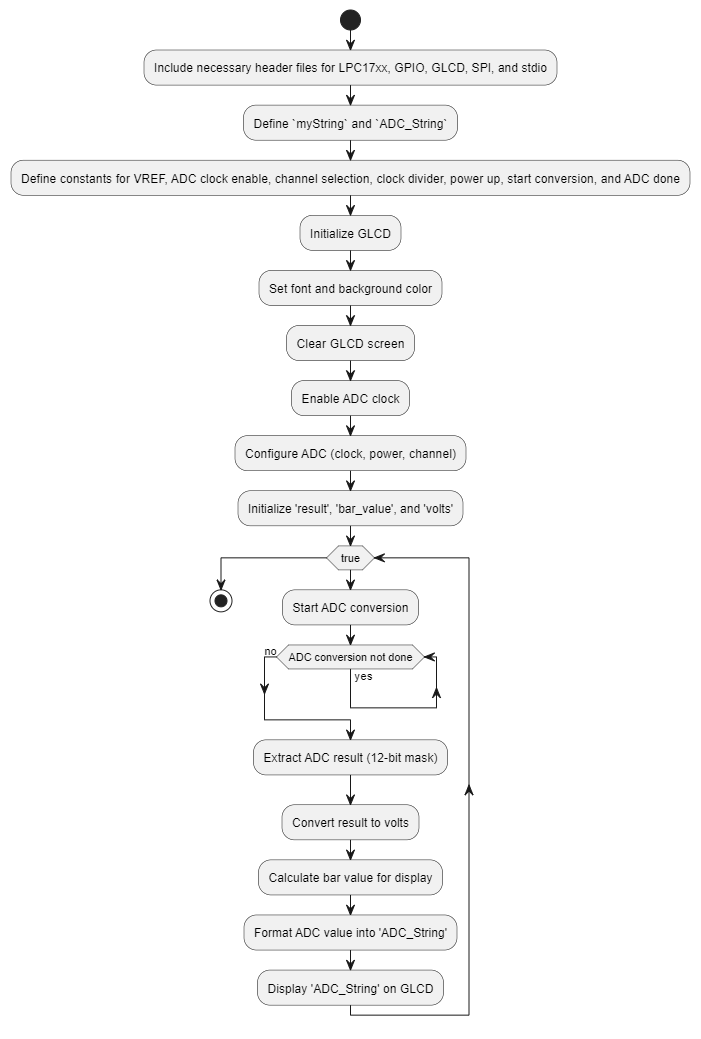
bar\_value = volts\*100/3.3;

sprintf(ADC\_String, "Volts: %f V", volts);

GLCD\_DrawString (5,200,ADC\_String);

}

}



### III.6 Lab Experiment 6 : Write the code to using Timer1 Interrupt to get data From ADC channel 2 using polling method then display 8-bit high value result to 8 bit LED with sampling time 1 second/1 time.

**CODE EXPLAINATION**

#include <LPC17xx.h>

#include <PIN\_LPC17xx.h>

#include <GPIO\_LPC17xx.h>

#include <GLCD\_Config.h>

#include <Board\_GLCD.h>

#include <GLCD\_Fonts.h>

#include <Driver\_SPI.h>

#include <stdio.h>

#define PRESCALE (25000-1) //25000 PCLK clock cycles to increment TC by 1

// Function prototypes

void initTimer0();

// String to be displayed on the LCD

char myString[] = {"hello, world"};

// String to store ADC conversion results

char ADC\_String[20] = " ";

// ADC configuration macros

#define VREF 3.3 // Reference Voltage at VREFP pin, given VREFN = 0V (GND)

#define ADC\_CLK\_EN (1<<12) // Enable ADC clock

#define SEL\_AD0\_2 (1<<2) // Select Channel AD0.2

#define CLKDIV 1 // ADC clock-divider (ADC\_CLOCK = PCLK / (CLKDIV + 1)) = 12.5 MHz for 25 MHz PCLK

#define PWRUP (1<<21) // Power up the ADC by setting this bit

#define START\_CNV (1<<24) // Start conversion immediately

#define ADC\_DONE (1U<<31) // Done bit to check if the ADC conversion is complete

#define ADCR\_SETUP\_SCM ((CLKDIV<<8) | PWRUP) // ADC control register setup

// Variables to store ADC result and converted voltage

int result = 0;

int bar\_value;

float volts = 0;

// Function to initialize ADC

void ADC\_init(void) {

LPC\_SC->PCONP |= ADC\_CLK\_EN; // Enable ADC clock

LPC\_ADC->ADCR = ADCR\_SETUP\_SCM | SEL\_AD0\_2; // Configure ADC settings and select AD0.2

LPC\_PINCON->PINSEL1 |= (1 << 18); // Select AD0.2 for P0.25

}

// Function to initialize the LCD display

void LCD\_disp\_init(void) {

GLCD\_Initialize(); // Initialize GLCD

GLCD\_SetFont(&GLCD\_Font\_16x24); // Set font for GLCD

GLCD\_SetBackgroundColor(GLCD\_COLOR\_GREEN); // Set background color to green

GLCD\_SetBackgroundColor(GLCD\_COLOR\_LIGHT\_GREY); // Set background color to light grey

GLCD\_ClearScreen(); // Clear the GLCD screen

}

// Function to display ADC result on the LCD

void ADC\_display(void) {

volts = (result \* VREF) / 4096.0; // Convert ADC result to voltage

bar\_value = volts \* 100 / 3.3; // Calculate bar value for display

sprintf(ADC\_String, "Volts: %f V", volts); // Format the voltage as a string

GLCD\_DrawString(5, 200, ADC\_String); // Display the voltage string on the GLCD

}

// Function to get ADC data

void ADC\_get\_data(void) {

LPC\_ADC->ADCR |= START\_CNV; // Start new ADC conversion

while ((LPC\_ADC->ADDR2 & ADC\_DONE) == 0); // Wait until conversion is finished

result = (LPC\_ADC->ADDR2 >> 4) & 0xFFF; // Extract the 12-bit ADC result

}

// Main function

int main(void) {

initTimer0(); // Initialize Timer0

ADC\_init(); // Initialize ADC

LCD\_disp\_init(); // Initialize LCD display

while (1) {

ADC\_display(); // Continuously display ADC result

}

}

// Function to initialize Timer0

void initTimer0(void) {

LPC\_SC->PCONP |= (1 << 1); // Power up TIM0. By default, TIM0 and TIM1 are enabled.

LPC\_SC->PCLKSEL0 &= ~(0x3 << 3); // Set PCLK for timer = CCLK / 4 = 100 / 4

LPC\_TIM0->CTCR = 0x0; // Set Timer0 to timer mode

LPC\_TIM0->PR = PRESCALE; // Set prescaler to divide PCLK by 50000

// 25000 clock cycles @ 25 MHz = 1 ms

LPC\_TIM0->MR0 = 2000; // Set match register for 2000 ms toggle time

LPC\_TIM0->MCR |= (1 << 0) | (1 << 1); // Interrupt and reset on MR0 match

LPC\_TIM0->TCR |= (1 << 1); // Reset Timer0

NVIC\_EnableIRQ(TIMER0\_IRQn); // Enable timer interrupt

LPC\_TIM0->TCR = 0x01; // Enable timer

}

// Timer0 interrupt handler

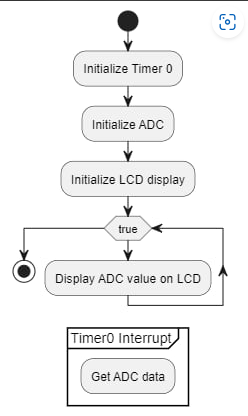
void TIMER0\_IRQHandler(void) {

LPC\_TIM0->IR |= (1 << 0); // Clear MR0 interrupt flag

ADC\_get\_data(); // Get ADC data in the interrupt

}

**FLOW CHART**



### III.7 Lab Experiment 7 : Write the code to using Timer1 Interrupt to get data From ADC channel 2 using polling method then display 12-bit value result to GLCD in form of text with sampling time 1 second/1 time.

### #include <LPC17xx.h>

### #include <PIN\_LPC17xx.h>

### #include <GPIO\_LPC17xx.h>

### #include <GLCD\_Config.h>

### #include <Board\_GLCD.h>

### #include <GLCD\_Fonts.h>

### #include <Driver\_SPI.h>

### #include <stdio.h>

### #define PRESCALE (50000-1) //25000 PCLK clock cycles to increment TC by 1

### #define PRESCALE1 (25000-1) //25000 PCLK clock cycles to increment TC by 1

### void initTimer0();

### char myString[]={"hello, world"};

### char ADC\_String[20]=" ";

### void delayms(unsigned int milliseconds);

### void Timer1\_Init(void);

### #define VREF 3.3 //Reference Voltage at VREFP pin, given VREFN = 0V(GND)

### #define ADC\_CLK\_EN (1<<12)

### #define SEL\_AD0\_2 (1<<2) //Select Channel AD0.0

### #define CLKDIV 1 //ADC clock-divider (ADC\_CLOCK=PCLK/CLKDIV+1) = 12.5Mhz

### //25Mhz PCLK

### #define PWRUP (1<<21) //setting it to 0 will power it down

### #define START\_CNV (1<<24) //001 for starting the conversion immediately

### #define ADC\_DONE (1U<<31) //define it as unsigned value Done Bit

### #define ADCR\_SETUP\_SCM ((CLKDIV<<8) | PWRUP)

### int result = 0;

### int bar\_value;

### float volts = 0;

### int main( void ) {

### LPC\_GPIO2->FIODIR |= (1<<2); //set P2.2 as output

### initTimer0();

### Timer1\_Init();

### GLCD\_Initialize() ;

### GLCD\_SetFont(&GLCD\_Font\_16x24) ;

### GLCD\_SetBackgroundColor (GLCD\_COLOR\_GREEN) ;

### GLCD\_SetBackgroundColor (GLCD\_COLOR\_WHITE);

### GLCD\_ClearScreen() ;

### 

### LPC\_SC->PCONP |= ADC\_CLK\_EN; //Enable ADC clock

### LPC\_ADC->ADCR = ADCR\_SETUP\_SCM | SEL\_AD0\_2;

### LPC\_PINCON->PINSEL1 |= (1<<18) ; //select AD0.2 for P0.25

### while ( 1 ) {

### GLCD\_SetForegroundColor (GLCD\_COLOR\_BLACK) ;

### GLCD\_DrawBargraph (10,100,bar\_value\*10,100,100);

### delayms(500);

### GLCD\_SetForegroundColor (GLCD\_COLOR\_WHITE) ;

### GLCD\_DrawBargraph (5,5,300,200,100);

### }

### }

### void initTimer0(void)

### {

### /\*Assuming that PLL0 has been setup with CCLK = 100Mhz and PCLK = 25Mhz.\*/

### LPC\_SC->PCONP |= (1<<1); //Power up TIM0. By default TIM0 and TIM1 are enabled.

### LPC\_SC->PCLKSEL0 &= ~(0x3<<3); //Set PCLK for timer = CCLK/4 = 100/4 (default)

### 

### LPC\_TIM0->CTCR = 0x0;

### LPC\_TIM0->PR = PRESCALE; //Increment LPC\_TIM0->TC at every 24999+1 clock cycles

### //25000 clock cycles @25Mhz = 1 mS

### 

### LPC\_TIM0->MR0 = 600; //Toggle Time in mS

### LPC\_TIM0->MCR |= (1<<0) | (1<<1); // Interrupt & Reset on MR0 match

### LPC\_TIM0->TCR |= (1<<1); //Reset Timer0

### NVIC\_EnableIRQ(TIMER0\_IRQn); //Enable timer interrupt

### 

### LPC\_TIM0->TCR = 0x01; //Enable timer

### }

### void TIMER0\_IRQHandler(void) //Use extern "C" so C++ can link it properly, for C it is not required

### {

### LPC\_TIM0->IR |= (1<<0); //Clear MR0 Interrupt flag

### LPC\_GPIO2->FIOPIN ^= (1<<2); //Toggle LED

### LPC\_ADC->ADCR |= START\_CNV; //Start new Conversion

### while((LPC\_ADC->ADDR2 & ADC\_DONE) == 0); //Wait untill conversion is

### //finished

### result = (LPC\_ADC->ADDR2>>4) & 0xFFF; //12 bit Mask to extract

### //result

### volts = (result\*VREF)/4096.0; //Convert result to Voltage

### bar\_value = volts\*100/3.3;

### 

### }

### void Timer1\_Init(void)

### {

### LPC\_SC->PCONP |= (1<<2); // turn on Timer1 module

### LPC\_SC->PCLKSEL0 &=(~(0x3<<5)); //0b010000; // systemcore clock/2= 100MHz/2 =50MHz

### 

### LPC\_TIM1->PR = PRESCALE1;

### LPC\_TIM1->TCR = 0x02;

### }

### 

### void delayms(unsigned int milliseconds)

### {

### LPC\_TIM1->TCR = 0x02; // reset counter b1 of TCR =1

### LPC\_TIM1->TCR = 0x01; // enable couter to count

### 

### while(LPC\_TIM1->TC < milliseconds);

### 

### LPC\_TIM1->TCR = 0x00;

### }

### 

### 

### III.8 Lab Experiment 7 : Write the code to using Timer1 Interrupt to get data From ADC channel 2 using polling method then display 12-bit value result to GLCD in form of bar graph with sampling time 1 second/1 time.

**CODE EXPLANATION**

/\*

\* leds.c

\*/

#include "LPC17xx.h"

#include "leds.h"

#include "adc.h"

#define POWER\_CONTROL (LPC\_SC->PCONP) // Power control for peripherals

#define PINCONSEL1 (LPC\_PINCON->PINSEL1) // Pin function select register 1

#define PINCONSEL2 (LPC\_PINCON->PINSEL2) // Pin function select register 2

#define IO1\_DIR (LPC\_GPIO1->FIODIR) // GPIO1 direction control register

#define IO2\_DIR (LPC\_GPIO2->FIODIR) // GPIO2 direction control register

#define IO1\_SET (LPC\_GPIO1->FIOSET) // GPIO1 set register

#define IO2\_SET (LPC\_GPIO2->FIOSET) // GPIO2 set register

#define IO1\_CLR (LPC\_GPIO1->FIOCLR) // GPIO1 clear register

#define IO2\_CLR (LPC\_GPIO2->FIOCLR) // GPIO2 clear register

#define IO1\_PIN (LPC\_GPIO1->FIOPIN) // GPIO1 pin state register

#define IO2\_PIN (LPC\_GPIO2->FIOPIN) // GPIO2 pin state register

// Define bit positions for each LED

#define LED1\_BIT (1 << 6)

#define LED2\_BIT (1 << 5)

#define LED3\_BIT (1 << 4)

#define LED4\_BIT (1 << 3)

#define LED5\_BIT (1 << 2)

#define LED6\_BIT (1 << 31)

#define LED7\_BIT (1 << 29)

#define LED8\_BIT (1 << 28)

// Function to initialize LEDs

void leds\_init(void)

{

POWER\_CONTROL |= (1 << 15); // Enable power for GPIO

IO2\_DIR = (LED1\_BIT | LED2\_BIT | LED3\_BIT | LED4\_BIT | LED5\_BIT); // Set GPIO2 direction for LEDs 1-5

IO1\_DIR = (LED6\_BIT | LED7\_BIT | LED8\_BIT); // Set GPIO1 direction for LEDs 6-8

}

// Function to turn on an LED

void led\_on(unsigned int index)

{

switch(index)

{

case 1:

IO2\_SET = LED1\_BIT; // Turn on LED1

break;

case 2:

IO2\_SET = LED2\_BIT; // Turn on LED2

break;

case 3:

IO2\_SET = LED3\_BIT; // Turn on LED3

break;

case 4:

IO2\_SET = LED4\_BIT; // Turn on LED4

break;

case 5:

IO2\_SET = LED5\_BIT; // Turn on LED5

break;

case 6:

IO1\_SET = LED6\_BIT; // Turn on LED6

break;

case 7:

IO1\_SET = LED7\_BIT; // Turn on LED7

break;

case 8:

IO1\_SET = LED8\_BIT; // Turn on LED8

break;

default:

break; // Default case does nothing

}

}

// Function to turn off an LED

void led\_off(unsigned int index)

{

switch(index)

{

case 1:

IO2\_CLR = LED1\_BIT; // Turn off LED1

break;

case 2:

IO2\_CLR = LED2\_BIT; // Turn off LED2

break;

case 3:

IO2\_CLR = LED3\_BIT; // Turn off LED3

break;

case 4:

IO2\_CLR = LED4\_BIT; // Turn off LED4

break;

case 5:

IO2\_CLR = LED5\_BIT; // Turn off LED5

break;

case 6:

IO1\_CLR = LED6\_BIT; // Turn off LED6

break;

case 7:

IO1\_CLR = LED7\_BIT; // Turn off LED7

break;

case 8:

IO1\_CLR = LED8\_BIT; // Turn off LED8

break;

default:

break; // Default case does nothing

}

}

// Function to set the LED value

void leds\_set\_value(char led\_Value)

{

int i;

for(i = 0; i < 8; i++)

{

if(led\_Value & (1 << i)) // Check if the i-th bit is set

{

led\_on(i + 1); // Turn on the corresponding LED

}

else

{

led\_off(i + 1); // Turn off the corresponding LED

}

}

}

/\*

\* main.c

\*/

int main()

{

leds\_init(); // Initialize LEDs

ADC\_Initialize(); // Initialize ADC

int ADC\_Value;

char led\_value;

while(1)

{

ADC\_StartConversion(); // Start ADC conversion

while(ADC\_ConversionDone() == 0); // Wait until conversion is done

ADC\_Value = ADC\_GetValue(); // Get ADC value

led\_value = (char) ADC\_Value; // Convert ADC value to char

leds\_set\_value(led\_value); // Set LED value

}

}

// Function to initialize Timer0

void initTimer0(void)

{

/\* Assuming that PLL0 has been setup with CCLK = 100Mhz and PCLK = 25Mhz. \*/

LPC\_SC->PCONP |= (1 << 1); // Power up TIM0. By default TIM0 and TIM1 are enabled.

LPC\_SC->PCLKSEL0 &= ~(0x3 << 3); // Set PCLK for timer = CCLK/4 = 100/4 (default)

LPC\_TIM0->CTCR = 0x0;

LPC\_TIM0->PR = PRESCALE; // Increment LPC\_TIM0->TC at every 24999+1 clock cycles

// 25000 clock cycles @25Mhz = 1 mS

LPC\_TIM0->MR0 = 2000; // Toggle Time in mS

LPC\_TIM0->MCR |= (1 << 0) | (1 << 1); // Interrupt & Reset on MR0 match

LPC\_TIM0->TCR |= (1 << 1); // Reset Timer0

NVIC\_EnableIRQ(TIMER0\_IRQn); // Enable timer interrupt

LPC\_TIM0->TCR = 0x01; // Enable timer

}

// Timer0 interrupt handler

void TIMER0\_IRQHandler(void) // Use extern "C" so C++ can link it properly, for C it is not required

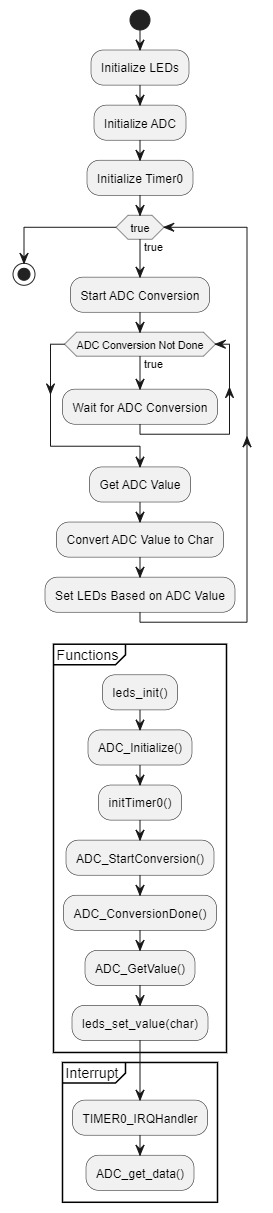
{

LPC\_TIM0->IR |= (1 << 0); // Clear MR0 Interrupt flag

ADC\_get\_data(); // Get ADC data

}

**FLOWCHART**



**IV. LAB PERFORMANCE GRADING AND LAB REPORT GUIDELINES**

For each Lab experiment Students show the successful running results to Lab Instructor for Lab Performance grading.

Students write a report which includes an algorithm flowchart and C++ Code for each experiment. In each block of the code or line of code, give the comments for the meaning of this block of code.

