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

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

Nguyễn Minh Đức – ITITIU21045

Tôn Long Thuận – ITITIU21322

### 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;

**#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=1.000.000/1.000 = 1000Hz=> T\_PCLK=1ms**

**void delayms(unsigned int milliseconds);**

**void Timer1\_Init(void);**

**char static myString[]={""};**

**int static x = 50;**

**int static y = 50;**

**int main( void ) {**

**Timer1\_Init();**

**GLCD\_Initialize();**

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

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

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

**GLCD\_ClearScreen();**

**while ( 1 ) {**

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

**delayms(20);**

**GLCD\_DrawRectangle((uint32\_t)x,(uint32\_t)y,100,100);**

**delayms(100);**

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

**GLCD\_DrawRectangle((uint32\_t)x,(uint32\_t)y,100,100);**

**if(x<=200 && y==10){**

**x+=10;**

**}**

**else if(x>200 && y<=120){**

**y+=10;**

**}**

**else if(x>10 && y>120){**

**x-=10;**

**}**

**else if(x==10){**

**y-=10;**

**}**

**}**

**}**

**void Timer1\_Init(void)**

**{**

**LPC\_SC->PCONP |= (1<<2); // turn on Timer3 module**

**LPC\_SC->PCLKSEL0 &= ~(0x3<<5) ; //0b010000; // systemcore clock/4= 100MHz/8 =25MHz**

**LPC\_TIM1->CTCR = 0x0;**

**LPC\_TIM1->PR = PRESCALE; // 25 000 000 / 25 000 = 1000 T = 1/f = =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 rectangle 100x100 pixel at position x=150, y=5; then move the rectangle in different position of the LCD screen with joystick controlled by four buttons P1.23,P1.24,P1.25,P1.26;

**#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 (50000-1)**

**void delayms(unsigned int milliseconds);**

**void Timer0\_Init(void);**

**#define Joystick\_key\_up 23**

**#define Joystick\_key\_down 24**

**#define Joystick\_key\_left 25**

**#define Joystick\_key\_right 26**

**int x=150;**

**int y=5;**

**int main( void ) {**

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

**Timer0\_Init();**

**GLCD\_Initialize() ;**

**GLCD\_ClearScreen();**

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

**GLCD\_SetBackgroundColor (GLCD\_COLOR\_BLACK) ;**

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

**GLCD\_DrawRectangle(x,y,100,100);**

**while ( 1 ) {**

**if((LPC\_GPIO1->FIOPIN & (1<<Joystick\_key\_up))==0)**

**{**

**delayms(100);**

**x=x+5;**

**}**

**if((LPC\_GPIO1->FIOPIN & (1<<Joystick\_key\_down))==0)**

**{**

**delayms(100);**

**x=x-5;**

**}**

**if((LPC\_GPIO1->FIOPIN & (1<<Joystick\_key\_left))==0)**

**{**

**delayms(100);**

**y=y+5;**

**}**

**if((LPC\_GPIO1->FIOPIN & (1<<Joystick\_key\_right))==0)**

**{**

**delayms(100);**

**y=y+5;**

**}**

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

**GLCD\_DrawRectangle((uint32\_t)x,(uint32\_t)y,100,100);**

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

**GLCD\_DrawRectangle((uint32\_t)x,(uint32\_t)y,100,100);**

**}**

**}**

**void Timer0\_Init(void)**

**{**

**LPC\_SC->PCONP |= (1<<23); // Turn on the power for timer3**

**LPC\_SC->PCLKSEL0 |= (1<<3); // PCLKSEL0[3:2]="10" mean system clock /2**

**// 100Mhz/2 = 50MHz/50.000 => 1000HHz => T=1ms**

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

**LPC\_TIM0->PR = PRESCALE;**

**LPC\_TIM0->TCR = 0x02;**

**}**

**void delayms(unsigned int milliseconds)**

**{**

**LPC\_TIM0->TCR = 0x02;**

**LPC\_TIM0->TCR = 0x01;**

**while(LPC\_TIM0->TC < milliseconds);**

**LPC\_TIM0->TCR = 0x00;**

**}**

### 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.

**#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=1.000.000/1.000 = 1000Hz=> T\_PCLK=1ms**

**void delayms(unsigned int milliseconds);**

**void Timer1\_Init(void);**

**char static myString[]={"Nguyen Minh Duc"};**

**int static x = 10;**

**int static y = 160;**

**int main( void ) {**

**Timer1\_Init();**

**GLCD\_Initialize();**

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

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

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

**GLCD\_ClearScreen();**

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

**delayms(20);**

**GLCD\_DrawString((uint32\_t)x,(uint32\_t)y,myString);**

**delayms(100);**

**GLCD\_SetBackgroundColor (GLCD\_COLOR\_BLUE);**

**GLCD\_Vscroll(200);**

**while ( 1 ) {**

**}**

**}**

**void Timer1\_Init(void)**

**{**

**LPC\_SC->PCONP |= (1<<2); // turn on Timer3 module**

**LPC\_SC->PCLKSEL0 &= ~(0x3<<5) ; //0b010000; // systemcore clock/4= 100MHz/8 =25MHz**

**LPC\_TIM1->CTCR = 0x0;**

**LPC\_TIM1->PR = PRESCALE; // 25 000 000 / 25 000 = 1000 T = 1/f = =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.4 Lab Experiment 4 : Write the C code to design the simple snake game. use GLCD Driver Library in GLCD to draw a rectangle 250x250 pixel at position x=1, y=1 ( considered as the wall). draw a rectangle 10x10 pixel at position x=20 y=20, withbgreen colors (considered as a snake); then move the snake to different position of the LCD screen with joystick controlled by four buttons P1.23,P1.24,P1.25,P1.26; When the snake hit the wall, stop the while loop and print out to the GLCD “you are lost”.

**#include <LPC17xx.h>**

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

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

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

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

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

**#include <stdlib.h>**

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

**#define xLim 200**

**#define yLim 120**

**#define PRESCALE (50000-1)**

**void delayms(unsigned int milliseconds);**

**void Timer0\_Init(void);**

**const static snakeSize = 5;**

**#define Joystick\_key\_up 23**

**#define Joystick\_key\_down 25**

**#define Joystick\_key\_left 26**

**#define Joystick\_key\_right 24**

**int static length = 3;**

**int static x[30] = {5, 10, 15, 20, 25, 30, 35, 40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};**

**int static y[30] = {5, 5, 5, 5, 5, 5, 5, 5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};**

**int static mode = 1;**

**int static xF = 0;**

**int static yF = 0;**

**int static state = 1;**

**char myString[]={"You're death !!!"};**

**void printSnakeB(){**

**for(int i =0; i<length;i++){**

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

**GLCD\_DrawRectangle((uint32\_t)x[i],(uint32\_t)y[i],snakeSize,snakeSize);**

**}**

**}**

**void rePrintFruit(){**

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

**GLCD\_DrawRectangle((uint32\_t)xF,(uint32\_t)yF,snakeSize,snakeSize);**

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

**GLCD\_DrawRectangle((uint32\_t)xF,(uint32\_t)yF,snakeSize,snakeSize);**

**}**

**void printSnakeW(){**

**for(int i =0; i<length;i++){**

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

**GLCD\_DrawRectangle((uint32\_t)x[i],(uint32\_t)y[i],snakeSize,snakeSize);**

**}**

**}**

**void printFruit(){**

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

**GLCD\_DrawRectangle((uint32\_t)xF,(uint32\_t)yF,snakeSize,snakeSize);**

**xF = (rand() % 36 )\*5;**

**yF = (rand() % 25 )\*5;**

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

**GLCD\_DrawRectangle((uint32\_t)xF,(uint32\_t)yF,snakeSize,snakeSize);**

**}**

**void updateSnake(int xNew, int yNew) {**

**printSnakeW();**

**for(int i=0; i<length-1;i++){**

**x[i] = x[i+1];**

**y[i] = y[i+1];**

**}**

**x[length-1] = xNew;**

**y[length-1] = yNew;**

**printSnakeB();**

**}**

**void checkDeath(){**

**if(x[length-1] > 315 || x[length-1] < 4 || y[length-1] >235 || y[length-1] < 0)**

**state =0;**

**}**

**void moveLeft(){**

**updateSnake(x[length-1]-5,y[length-1]);**

**}**

**void moveRight(){**

**updateSnake(x[length-1]+5,y[length-1]);**

**}**

**void moveUp(){**

**updateSnake(x[length-1],y[length-1]-5);**

**}**

**void moveDown(){**

**updateSnake(x[length-1],y[length-1]+5);**

**}**

**void move(){**

**switch (mode){**

**case 0:**

**moveLeft();**

**break;**

**case 1:**

**moveRight();**

**break;**

**case 2:**

**moveUp();**

**break;**

**case 3:**

**moveDown();**

**break;**

**default:**

**break;**

**}**

**}**

**void checkEat(){**

**for (int i =0; i<length;i++){**

**if (xF == x[i] && yF == y[i]){**

**printFruit();**

**x[length] = x[length-1];**

**y[length] = y[length-1];**

**length++;**

**}**

**}**

**}**

**int main( void ) {**

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

**Timer0\_Init();**

**GLCD\_Initialize() ;**

**GLCD\_ClearScreen();**

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

**GLCD\_SetBackgroundColor (GLCD\_COLOR\_BLACK) ;**

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

**printFruit();**

**while ( state ) {**

**for (int i =0; i <10 ; i++){**

**if (i==5){**

**move();**

**rePrintFruit();**

**checkEat();}**

**delayms(10);**

**checkDeath();**

**if((LPC\_GPIO1->FIOPIN & (1<<Joystick\_key\_up))==0){**

**delayms(10);**

**if (mode == 3) mode =3; else mode =2;**

**delayms(10);**

**}**

**if((LPC\_GPIO1->FIOPIN & (1<<Joystick\_key\_down))==0){**

**delayms(10);**

**if (mode == 2) mode =2; else mode =3;**

**delayms(10);**

**}**

**if((LPC\_GPIO1->FIOPIN & (1<<Joystick\_key\_left))==0){**

**delayms(10);**

**if (mode == 1) mode =1; else mode =0;**

**delayms(10);**

**}**

**if((LPC\_GPIO1->FIOPIN & (1<<Joystick\_key\_right))==0){**

**delayms(10);**

**if (mode == 0) mode =0; else mode =1;**

**delayms(10);**

**}**

**if (state == 0)**

**GLCD\_DrawString (5,200,myString);**

**}**

**}**

**}**

**void Timer0\_Init(void)**

**{**

**LPC\_SC->PCONP |= (1<<23); // Turn on the power for timer3**

**LPC\_SC->PCLKSEL0 |= (1<<3); // PCLKSEL0[3:2]="10" mean system clock /2**

**// 100Mhz/2 = 50MHz/50.000 => 1000HHz => T=1ms**

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

**LPC\_TIM0->PR = PRESCALE;**

**LPC\_TIM0->TCR = 0x02;**

**}**

**void delayms(unsigned int milliseconds)**

**{**

**LPC\_TIM0->TCR = 0x02;**

**LPC\_TIM0->TCR = 0x01;**

**while(LPC\_TIM0->TC < milliseconds);**

**LPC\_TIM0->TCR = 0x00;**

**}**

### III.5 Lab Experiment 5 : Write the C code to get data From ADC channel 2 using polling method then display 12-bit result to text string displayed in GLCD of the Experiment KIT .

**#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<<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 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->PINSEL1 |= (1<<18) ; //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->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;**

**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 then display 12-bit result to text string displayed in GLCD of the Experiment KIT with sampling time 2 second interval.

**#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**

**void initTimer0();**

**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<<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;**

**void ADC\_init(void)**

**{**

**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**

**}**

**void LCD\_disp\_init(void)**

**{**

**GLCD\_Initialize() ;**

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

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

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

**GLCD\_ClearScreen() ;**

**}**

**void ADC\_display(void)**

**{**

**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);**

**}**

**void ADC\_get\_data(void)**

**{**

**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**

**}**

**int main( void ) {**

**initTimer0();**

**ADC\_init();**

**LCD\_disp\_init();**

**while ( 1 )**

**{**

**ADC\_display();**

**}**

**}**

**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**

**}**

**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();**

**}**

### 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 bar graph 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 8 : 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.

**include "LPC17xx.h"**

**#define POWER\_CONTROL (LPC\_SC->PCONP)**

**#define PINCONSEL1 (LPC\_PINCON->PINSEL1)**

**#define PINCONSEL2 (LPC\_PINCON->PINSEL2)**

**#define IO1\_DIR (LPC\_GPIO1->FIODIR)**

**#define IO2\_DIR (LPC\_GPIO2->FIODIR)**

**#define IO1\_SET (LPC\_GPIO1->FIOSET)**

**#define IO2\_SET (LPC\_GPIO2->FIOSET)**

**#define IO1\_CLR (LPC\_GPIO1->FIOCLR)**

**#define IO2\_CLR (LPC\_GPIO2->FIOCLR)**

**#define IO1\_PIN (LPC\_GPIO1->FIOPIN)**

**#define IO2\_PIN (LPC\_GPIO2->FIOPIN)**

**#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)**

**#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)**

**void leds\_init(void)**

**{**

**POWER\_CONTROL |= (1 << 15);**

**IO2\_DIR = (LED1\_BIT | LED2\_BIT | LED3\_BIT | LED4\_BIT | LED5\_BIT);**

**IO1\_DIR = (LED6\_BIT | LED7\_BIT | LED8\_BIT);**

**}**

**void led\_on(unsigned int index)**

**{**

**switch(index)**

**{**

**case 1:**

**IO2\_SET = LED1\_BIT;**

**break;**

**case 2:**

**IO2\_SET = LED2\_BIT;**

**break;**

**case 3:**

**IO2\_SET = LED3\_BIT;**

**break;**

**case 4:**

**IO2\_SET = LED4\_BIT;**

**break;**

**case 5:**

**IO2\_SET = LED5\_BIT;**

**break;**

**case 6:**

**IO1\_SET = LED6\_BIT;**

**break;**

**case 7:**

**IO1\_SET = LED7\_BIT;**

**break;**

**case 8:**

**IO1\_SET = LED8\_BIT;**

**break;**

**default:**

**break;**

**}**

**}**

**void led\_off(unsigned int index)**

**{**

**switch(index)**

**{**

**case 1:**

**IO2\_CLR = LED1\_BIT;**

**break;**

**case 2:**

**IO2\_CLR = LED2\_BIT;**

**break;**

**case 3:**

**IO2\_CLR = LED3\_BIT;**

**break;**

**case 4:**

**IO2\_CLR = LED4\_BIT;**

**break;**

**case 5:**

**IO2\_CLR = LED5\_BIT;**

**break;**

**case 6:**

**IO1\_CLR = LED6\_BIT;**

**break;**

**case 7:**

**IO1\_CLR = LED7\_BIT;**

**break;**

**case 8:**

**IO1\_CLR = LED8\_BIT;**

**break;**

**default:**

**break;**

**}**

**}**

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

**{**

**int i;**

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

**{**

**if(led\_Value & (1<<i))**

**{**

**led\_on(i+1);**

**}**

**else**

**{**

**led\_off(i+1);**

**}**

**}**

**}**

**void ADC\_Initialize(void)**

**{**

**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**

**}**

**void ADC\_StartConversion(void)**

**{**

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

**}**

**int main()**

**{**

**leds\_init();**

**ADC\_Initialize();**

**int ADC\_Value;**

**char led\_value;**

**while(1)**

**{**

**ADC\_StartConversion();**

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

**//finished**

**ADC\_Value = (LPC\_ADC->ADDR2>>8) & 0xFF; //12 bit Mask to extract**

**//result**

**led\_value = (char) ADC\_Value;**

**leds\_set\_value(led\_value);**

**}**

}

**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 : 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.