LAB NO : 10 DATE : 19/03/2025

Title : ANALOG TO DIGITAL CONVERTOR PROGRAM

**AN\_LCD.h**

#include <lpc17xx.h>

void clear\_ports(void){

/\* Clearing the lines at power on \*/

LPC\_GPIO0->FIOCLR = 0x0F << 23; // Clearing data lines

LPC\_GPIO0->FIOCLR = 1 << 27; // Clearing RS line

LPC\_GPIO0->FIOCLR = 1 << 28; // Clearing Enable line

return;

}

void delay\_lcd(unsigned int r1){

unsigned int r;

for(r = 0; r < r1; r++);

return;

}

void write(int temp2, int type){ // Write to command/data register

clear\_ports();

LPC\_GPIO0->FIOPIN = temp2; // Assign the value to the data lines

if(type == 0)

LPC\_GPIO0->FIOCLR = 1 << 27; // Clear bit RS for Command

else

LPC\_GPIO0->FIOSET = 1 << 27; // Set bit RS for Data

LPC\_GPIO0->FIOSET = 1 << 28; // EN = 1

delay\_lcd(25);

LPC\_GPIO0->FIOCLR = 1 << 28; // EN = 0

return;

}

void lcd\_comdata(int temp1, int type){

int temp2 = temp1 & 0xf0; // Move data (26-8+1) times : 26 - HN place, 4 - Bits

temp2 = temp2 << 19; // Data lines from 23 to 26

write(temp2, type);

temp2 = temp1 & 0x0f; // 26-4+1

temp2 = temp2 << 23;

write(temp2, type);

delay\_lcd(1000);

return;

}

void lcd\_init(){

/\* Ports initialized as GPIO \*/

LPC\_PINCON->PINSEL1 &= 0xFC003FFF; // P0.23 to P0.28

/\* Setting the directions as output \*/

LPC\_GPIO0->FIODIR |= 0x0F<<23 | 1<<27 | 1<<28;

clear\_ports();

delay\_lcd(3200);

lcd\_comdata(0x33, 0);

delay\_lcd(30000);

lcd\_comdata(0x32, 0);

delay\_lcd(30000);

lcd\_comdata(0x28, 0); // Function set

delay\_lcd(30000);

lcd\_comdata(0x0c, 0); // Display on, cursor off

delay\_lcd(800);

lcd\_comdata(0x06, 0); // Entry mode set, increment cursor right

delay\_lcd(800);

lcd\_comdata(0x01, 0); // Display clear

delay\_lcd(10000);

return;

}

void lcd\_puts(unsigned char \*buf1){

unsigned int i = 0;

unsigned int temp3;

while(buf1[i] != '\0'){

temp3 = buf1[i];

lcd\_comdata(temp3, 1);

i++;

if(i == 16)

lcd\_comdata(0xc0, 0); // Move to the next line on LCD

}

return;

}

# Solved Exercise 1: WAP to configure and read analog data from ADC channel no 5, and display the digital data on the LCD.

## Code :

#include<LPC17xx.h>

#include<stdio.h>

#include"AN\_LCD.h"

#define Ref\_Vtg 3.300 //Reference voltage (3.3V for the ADC)

#define Full\_Scale 0xFFF //Full scale ( maximum possible ADC value) for a 12-bit ADC (4095 in decimal)

int main(void) {

unsigned long adc\_temp; //hold the ADC conversion result

unsigned int i;

float in\_vtg; //stores the calculated input voltage

unsigned char vtg[7], dval[7]; //hold strings for displaying voltage and ADC value

unsigned char Msg3[11] = {"ANALOG IP:"};

unsigned char Msg4[12] = {"ADC OUTPUT:"};

SystemInit();

SystemCoreClockUpdate();

LPC\_SC->PCONP |= (1<<15); //Power for GPIO block

lcd\_init();

LPC\_PINCON->PINSEL3 |= 0xC0000000; //P1.31 as AD0.5

LPC\_SC->PCONP |= (1<<12); //enable the peripheral ADC

SystemCoreClockUpdate();

lcd\_comdata(0x80, 0);

delay\_lcd(800);

lcd\_puts(&Msg3[0]);

lcd\_comdata(0xC0, 0);

delay\_lcd(800);

lcd\_puts(&Msg4[0]);

while(1){

LPC\_ADC->ADCR = (1<<5)|(1<<21)|(1<<24); //0x01200001;

//ADC0.5, start conversion and operational

while(!(LPC\_ADC->ADDR5 & 0x80000000));

//wait till 'done' bit is 1, indicates conversion complete

adc\_temp = LPC\_ADC->ADDR5;

adc\_temp >>= 4;

adc\_temp &= 0x00000FFF; //12 bit ADC

in\_vtg = (((float)adc\_temp \* (float)Ref\_Vtg))/((float)Full\_Scale);

//calculating input analog voltage

sprintf(vtg, "%3.2fV", in\_vtg);

//convert the readings into string to display on LCD

sprintf(dval, "%x", adc\_temp);

for(i=0; i<2000; i++);

lcd\_comdata(0x89, 0);

delay\_lcd(800);

lcd\_puts(&vtg[0]);

lcd\_comdata(0xC8, 0);

delay\_lcd(800);

lcd\_puts(&dval[0]);

for(i=0;i<200000;i++);

for(i=0;i<7;i++)

vtg[i] = dval[i] = 0x00;

adc\_temp = 0;

in\_vtg = 0;

}

}

# Exercise 1: Write a C program to display the digital value representing the difference in analog voltages at ADC channel 4 and channel 5 on LCD using BURST and Software mode.

## Code :

#include <LPC17xx.h>

#include <stdio.h>

#include "AN\_LCD.h"

#define Ref\_Vtg 3.300 // Reference voltage (3.3V)

#define Full\_Scale 0xFFF // 12-bit ADC scale (4095 in decimal)

int main(void){

unsigned long adc\_temp1, adc\_temp2;

unsigned int i;

float in\_vtg1, in\_vtg2, in\_vdiff;

unsigned char vtg1[6], vtg2[6], vdiff[6];

unsigned char Msg3[] = {"CH4,5:"};

unsigned char Msg4[] = {"V Diff: "};

SystemInit();

SystemCoreClockUpdate();

LPC\_SC->PCONP |= (1 << 15); // Power for GPIO block

lcd\_init();

LPC\_PINCON->PINSEL3 |= 0x30000000; // P1.30 as AD0.4

LPC\_PINCON->PINSEL3 |= 0xC0000000; // P1.31 as AD0.5

LPC\_SC->PCONP |= (1 << 12); // enable the peripheral ADC

SystemCoreClockUpdate();

//Displaying CH4,5: in line 1 at the beginning

lcd\_comdata(0x80, 0);

delay\_lcd(800);

lcd\_puts(&Msg3[0]);

//Displaying V Diff: in line 2 at the beginning

lcd\_comdata(0xC0, 0);

delay\_lcd(800);

lcd\_puts(&Msg4[0]);

// Burst mode and software trigger configuration

// Burst mode - ADC performs multiple conversions in rapid succession without requiring a //separate trigger for each

LPC\_ADC->ADCR = (1 << 4) | (1 << 5) | (1 << 21) | (1 << 16);

//CH4, CH5, //Enables ADC clock, starts the ADC conversion using software triggering

while (1){

// wait till 'done' bit is 1, indicates conversion complete

while (!(LPC\_ADC->ADDR5 & (1 << 31)));

adc\_temp1 = LPC\_ADC->ADDR4; //reads the ADC result from ch4

adc\_temp1 >>= 4; //Data is stored from 4th bit onwards => right shift

adc\_temp1 &= 0xFFF; //Extracting the 12-bit ADC data

adc\_temp2 = LPC\_ADC->ADDR5;

adc\_temp2 >>= 4;

adc\_temp2 &= 0x00000FFF; // 12 bit ADC

//Calculating input analog voltage and their difference

in\_vtg1 = (((float)adc\_temp1 \* (float)Ref\_Vtg)) / ((float)Full\_Scale);

in\_vtg2 = (((float)adc\_temp2 \* (float)Ref\_Vtg)) / ((float)Full\_Scale);

in\_vdiff = in\_vtg1 - in\_vtg2;

//Converting value into a string

sprintf(vtg1, "%3.2f", in\_vtg1);

sprintf(vtg2, "%3.2f", in\_vtg2);

sprintf(vdiff, "%3.2f", in\_vdiff);

for (i = 0; i < 2000; i++); //Delay

lcd\_comdata(0x86, 0); //Displaying V1 in line 1 after string

delay\_lcd(800);

lcd\_puts(&vtg1[0]);

lcd\_comdata(0x8B, 0); //Displaying V2 in line 1 after V1

delay\_lcd(800);

lcd\_puts(&vtg2[0]);

lcd\_comdata(0xC8, 0); //Displaying V1-V2 in line 2 after string

delay\_lcd(800);

lcd\_puts(&vdiff[0]);

for (i = 0; i < 200000; i++); //Delay

//Resetting

for (i = 0; i < 7; i++)

vtg1[i] = vtg2[i] = 0;

adc\_temp1 = 0;

in\_vtg1 = 0;

adc\_temp2 = 0;

in\_vtg2 = 0;

}

}