Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of ECE
Program – UG – 2024 – Semester VI
Embedded System Design Lab – ECL66

ESD LAB PROGRAMS

Header files

1. `stdio.h` - it is used for `printf()` function
2. `sys/types.h` - it is used for `pid_t` type, that is the data type of the variables which are using to store the process ids.
3. `unistd.h` - it is used for `getpid()` and `getppid()` functions, fork( ), read( ), write ( )
4. stdlib.h – it is used for atoi( ) function
5. fcntl.h – it is used for open( ), close( ) functions

Commands for Linux:

gedit filename.c	**// to edit file**
gcc –o filename filename.c	**// to compile the file**
./filename	**// to run the executable file**

Write a C program to demonstrate usage of fork.

```
 #include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main() {
   fork();
   printf("Fork testing code\n");
   return 0;
}
```

Output:

```
Fork testing code
Fork testing code
```

Write a C program to demonstrate usage of fork with child and parent process ID is printed and parent waits for child process to terminate.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>
int
main (void)
{
 pid_t pid;
 char *message;
 int no, NO1 = 1;
 int i, l;
 printf ("calling fork \n");
 pid = fork ();

  switch (pid)
   {
   case -1:
     printf ("fork failed \n");
     exit (1);

   case 0:
     message = "Child Process";
     i = 1;
     no = getpid ();
     NO1 = getppid ();
     break;

   default:
     message = "Parent Process";
     i = 1;

      no = getpid ();
      NO1 = getppid ();
      break;
    }

    if(pid !=0) {
    printf("HP: hello from parent\n");
       wait(NULL);
       printf("CT: child has terminated\n");
    }

  for (l = i; l > 0; l--)
    {
      puts (message);
      printf ("My ID is %d \n", no);
      printf ("My parent ID is %d \n", NO1);
         }
  return (0);
}
```

Usage of “Signal” function calls–

Write a C program to demonstrate usage of signal function calls: when CTRL C is pressed a signal is sent for abrupt termination.

```
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
void my_handler(int signal)
{
printf("Problem encountered %d \n", signal);
}
int main()
{
(void) signal (SIGINT,my_handler);
while(1)
{
printf("Hello \n");
sleep(2);
}
}
```

Write a C program to demonstrate usage of signal function calls: when CTRL C is pressed a signal ignore the signal.

```
#include <signal.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
(void) signal (SIGINT,SIG_IGN);
while(1)
{
printf("%d \n", getpid());
sleep(1);
}
}
```

Multithreading

gcc –o filename filename.c   -lpthread   // to compile the file

Write a C program for : One thread reads the input from the keyboard and another thread converts to upper case. This is done until Stop" is pressed. Use concept of multithreading

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include<string.h>
#include<ctype.h>
#include <pthread.h>

#define BUFFER_SIZE 1024

char buffer[BUFFER_SIZE];

void *read_thread (void *arg)
{
  while (strncmp ("stop", buffer, 4) != 0)
    {
      printf ("Enter text:  ");
      fgets (buffer, BUFFER_SIZE, stdin);
      sleep (1);
    }

  pthread_exit ("read_thread exit successful");
 }

void *convert_thread ()
{
  int i;
  while (strncmp ("stop", buffer, 4) != 0)
    {
      sleep (1);
      printf ("Converted text: ");
      for (i = 0; i < strlen (buffer); i++)
            printf ("%c", toupper (buffer[i]));
    }
  pthread_exit ("convert_thread exit successful");
}

int main ()
{
  int result;
  pthread_t rthread, cthread;
  void *thread_result;
  printf ("Enter text, the program will convert it into upper case, \n To stop enter 'stop' n");
  pthread_create (&rthread, NULL, read_thread, NULL);
  pthread_create (&cthread, NULL, convert_thread, NULL);

  pthread_join (rthread, &thread_result);
  printf ("read_thread joined, %s\n",(char)thread_result);
  pthread_join(cthread, &thread_result);
  printf ("convert_thread joined, %s\n", (char *) thread_result);
return(0);
}
```

Intertask communication using semaphore

Write a C program for : One thread reads the input from the keyboard and another thread converts to upper case. This is done until Stop" is pressed. Use concept of semaphore, so that multiple printing is avoided.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include<string.h>
#include <ctype.h>
#include <pthread.h>
#include <semaphore.h>

#define BUFFER_SIZE 1024
sem_t sem;
char buffer[BUFFER_SIZE];

void *read_thread(void *arg)
{
while(strncmp("stop",buffer,4) != 0)
{
printf("Enter text:  ");
fgets(buffer, BUFFER_SIZE, stdin);
sem_post(&sem);
printf("%d \n",sem);
sleep(2);
}

pthread_exit("read_thread exit successful");
}

void *convert_thread()
{
int i; sem_wait(&sem);
while(strncmp("stop", buffer, 4) != 0)
{
printf("Converted text: ");
for(i=0; i<strlen(buffer); i++)
printf("%c", toupper(buffer[i]));
sem_wait(&sem);
}
pthread_exit("convert_thread exit successful");
}
```

```
int main()
{
int result;
pthread_t rthread, cthread;
void *thread_result;
sem_init(&sem, 0, 1);
printf("Enter text, the program will convert it into upper case, \n To stop enter 'stop' \n");
pthread_create(&cthread, NULL, convert_thread, NULL);
pthread_create(&rthread, NULL, read_thread, NULL);
result = pthread_join(rthread, &thread_result);
printf("read_thread joined, %s\n",(char *)thread_result);
pthread_join(cthread, &thread_result);
printf("convert_thread joined, %s\n",(char *)thread_result);
sem_destroy(&sem);
exit(0);
}
```

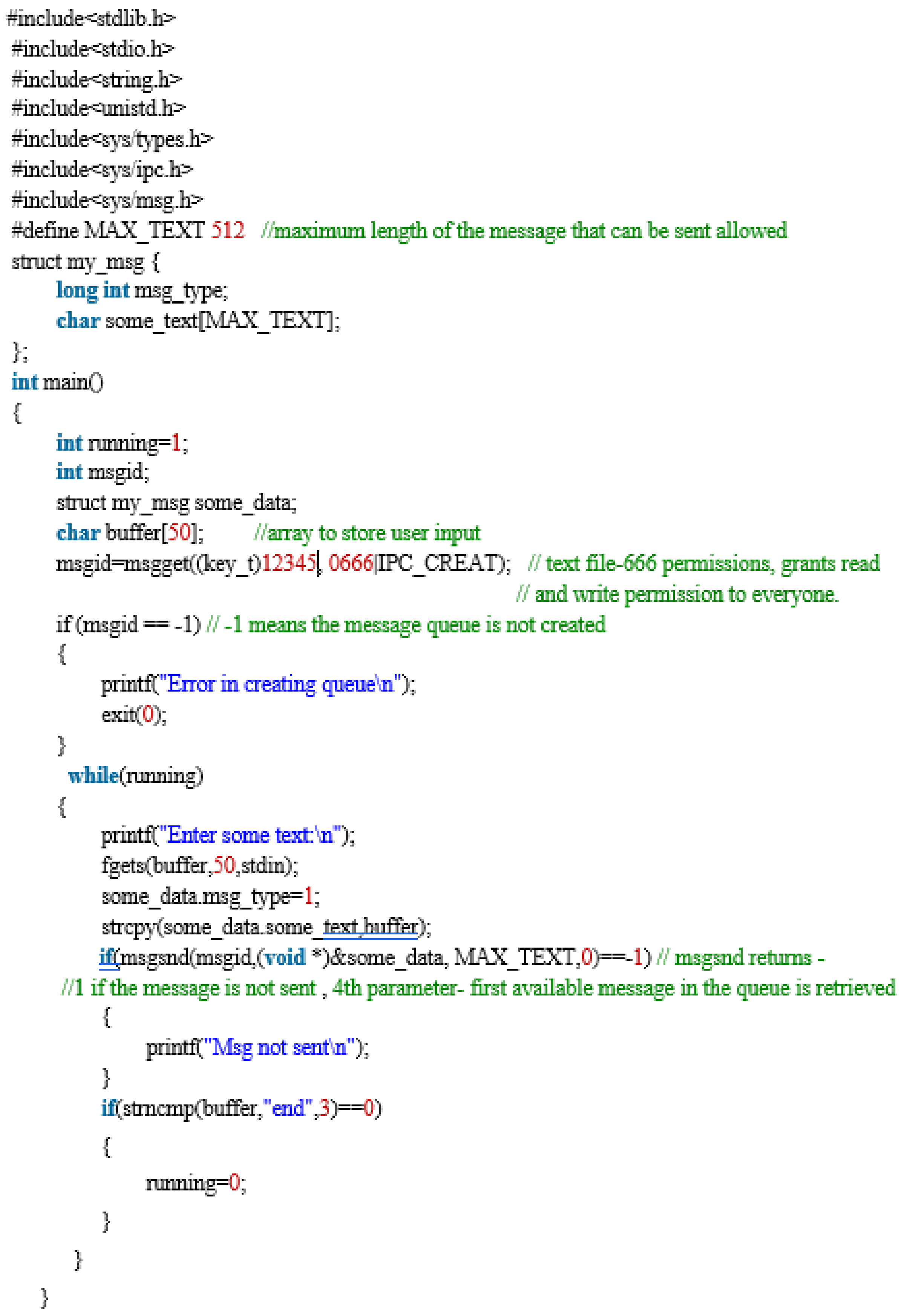
Intertask communication using mutex

Write a C program using pthreads that demonstrates the use of a mutex to synchronize access to a shared variable by two threads. One thread should increment the variable, while the other thread should decrement it.

```
#include <stdio.h>
#include <pthread.h>
#define MAX_COUNT 10
int counter = 0;
pthread_mutex_t mutex;

void *up_counter(void *arg)
 {
   for (int i = 0; i < MAX_COUNT; ++i)
 {
       pthread_mutex_lock(&mutex);
       counter++;
       printf("Upcount: %d\n", counter);
       pthread_mutex_unlock(&mutex);
   }
   pthread_exit(NULL);
  }

void *down_counter(void *arg)
 {
   for (int i = 0; i < MAX_COUNT; ++i)
 {
       pthread_mutex_lock(&mutex);
       counter--;
       printf("Downcount: %d\n", counter);
       pthread_mutex_unlock(&mutex);
   }
   pthread_exit(NULL);
}

int main()
{
   pthread_t thread_up, thread_down;

   // Initialize mutex
   pthread_mutex_init(&mutex, NULL);     //  2nd attribute-default, non recursive is used

   // Create threads
   pthread_create(&thread_up, NULL, up_counter, NULL);
   pthread_create(&thread_down, NULL, down_counter, NULL);

   // Join threads
   pthread_join(thread_up, NULL);
   pthread_join(thread_down, NULL);

   // Destroy mutex
   pthread_mutex_destroy(&mutex);
   return 0;
  }
```

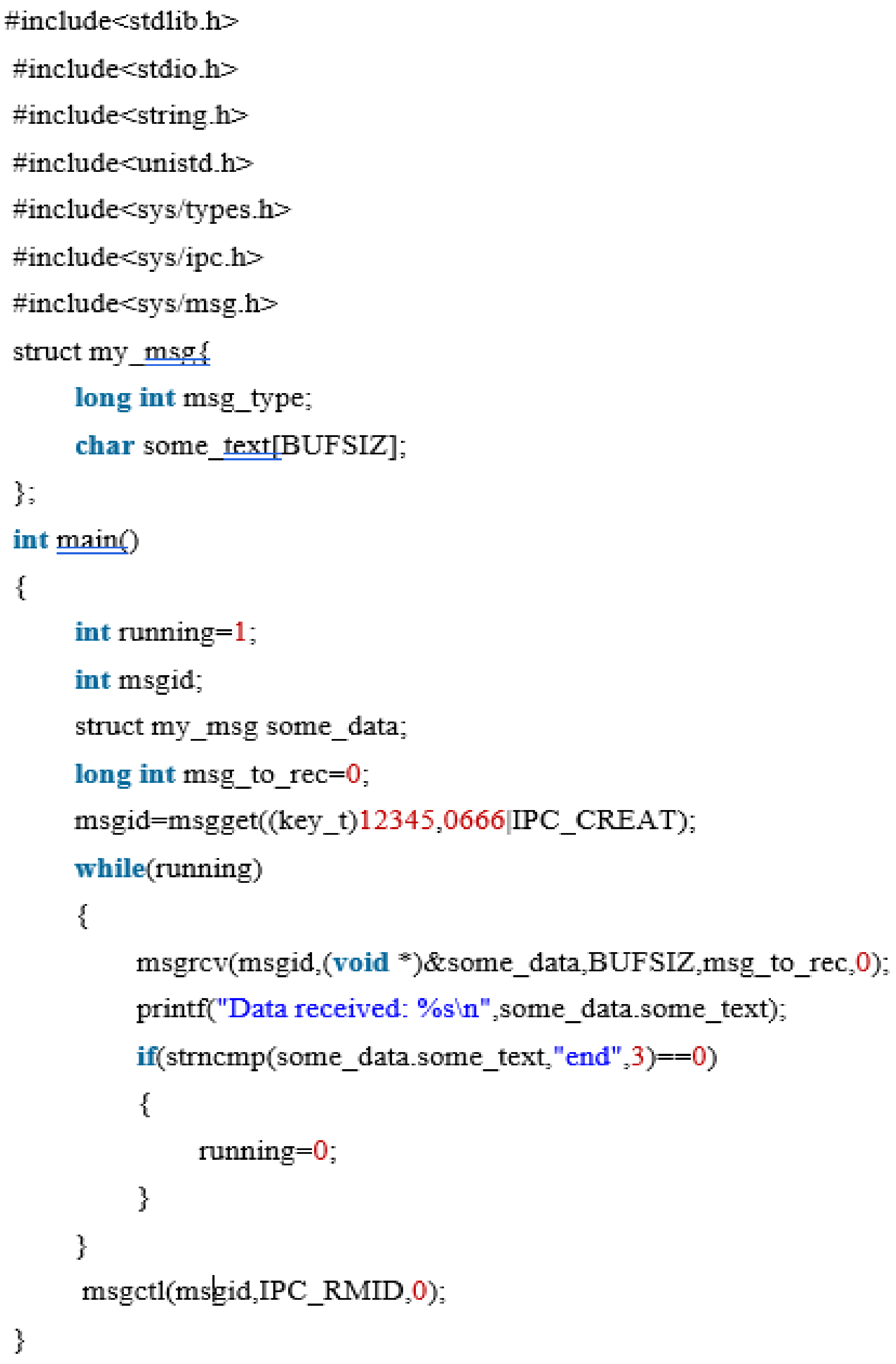
Interprocess communication using Message Queue

Program i) Write a program for IPC using Message Queues to send data to a message queue



```
#include<stdlib.h>
 #include<stdio.h>
 #include<string.h>
 #include<unistd.h>
 #include<sys/types.h>
 #include<sys/ipc.h>
 #include<sys/msg.h>
 #define MAX_TEXT 512   //maximum length of the message that can be sent allowed
 struct my_msg {
        long int msg_type;
        char some_text[MAX_TEXT];
 };
 int main()
 {
        int running=1;
        int msgid;
        struct my_msg some_data;
        char buffer[50];        //array to store user input
        msgid=msgget((key_t)12345, 0666|IPC_CREAT);   // text file-666 permissions, grants read
                                                     // and write permission to everyone.
        if (msgid == -1) // -1 means the message queue is not created
        {
               printf("Error in creating queue\n");
               exit(0);
        }
          while(running)
        {
               printf("Enter some text:\n");
               fgets(buffer,50,stdin);
               some_data.msg_type=1;
               strcpy(some_data.some_text,buffer);
               if(msgsnd(msgid,(void *)&some_data, MAX_TEXT,0)==-1) // msgsnd returns -
        //1 if the message is not sent , 4th parameter- first available message in the queue is retrieved
               {
                      printf("Msg not sent\n");
               }
               if(strncmp(buffer,"end",3)==0)
               {
                      running=0;
               }
           }
     }
```

Program ii) Write a program for IPC using Message Queues to receive or read message from the above created message queue(Created in Program 1).

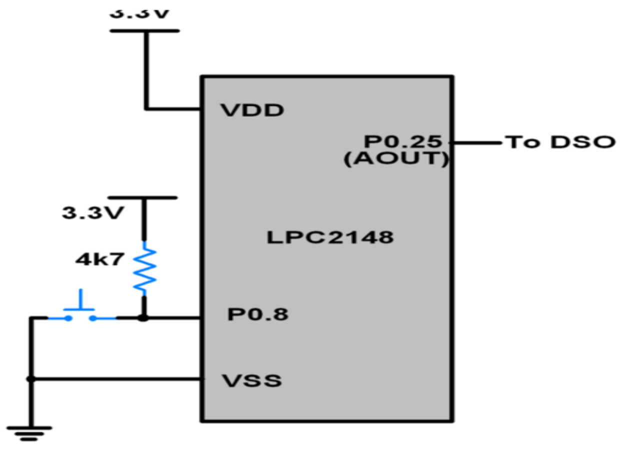


```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>
struct my_msg{
        long int msg_type;
        char some_text[BUFSIZ];
};
int main()
{
        int running=1;
        int msgid;
        struct my_msg some_data;
        long int msg_to_rec=0;
        msgid=msgget((key_t)12345,0666|IPC_CREAT);
        while(running)
        {
                msgrcv(msgid,(void *)&some_data,BUFSIZ,msg_to_rec,0);
                printf("Data received: %s\n",some_data.some_text);
                if(strncmp(some_data.some_text,"end",3)==0)
                {
                        running=0;
                }
        }
         msgctl(msgid,IPC_RMID,0);
}
```

INTERFACE PROGRAMS

Waveform generation using the internal DAC of LPC2148

Waveform generation using the internal DAC of LPC 2148.



Program (1)

//Sine

```
#include "lpc214x.h"
#include "stdint.h"
 voiddelay_ms(uint16_zt j)
{
uint16_tx,i;
        for(i=0;i<j; i++)
        {
for(x=0; x<6000; x++); /* loop to generate 1 milisecond delay with Cclk =60MHz */
        }
}
int main (void)
{
uint16_t value;
uint16_ti = 0;
uint16_t sintable[64]={512,562,611,660,707,753,796,836,873,907,937,963,984,
1001,1013,1021,1023,1021,1013,1001,984,963,937,907,873,836,796,753,707,660,
611,562,512,461,412,363,316,270,227,187,150,116,86,60,39,22,10,2,0,2,10,22,39,
60,86,116,150,187,227,270,316,363,412,461};

        PINSEL1 = 0x00080000;/* P0.25 as DAC output */
        IO0DIR = 0xFFFFFFFF;
        while(1)
        {
        while(i<64)
        {
        value=(sintable[i]<<6);
        DACR=value;
        delay_ms(1);
        i++;
        }
        i=0;
      }
     }
```

Program (2)

//triangle

```
#include "lpc214x.h"
#include "stdint.h"
voiddelay_ms(uint16_t j)
{
 uint16_tx, i;
        for(i=0; i<j; i++)
        {
        for(x=0; x<6000; x++);        /* loop to generate delay */
        }
 }

 int main (void)
 {
 uint16_t value;
 uint16_ti = 0;
 PINSEL1 = 0x00080000;        /* P0.25 as DAC output */
 IO0DIR = 0xFFFFFFFF
        while(1)
        {
        i=0;
        while(i!=1023)
        {
        DACR=i<<6;
        i++;
        }
        i=1023;
        while(i!=0)
        {
        DACR=i<<6;
        i--;
        }
        }
    }
```

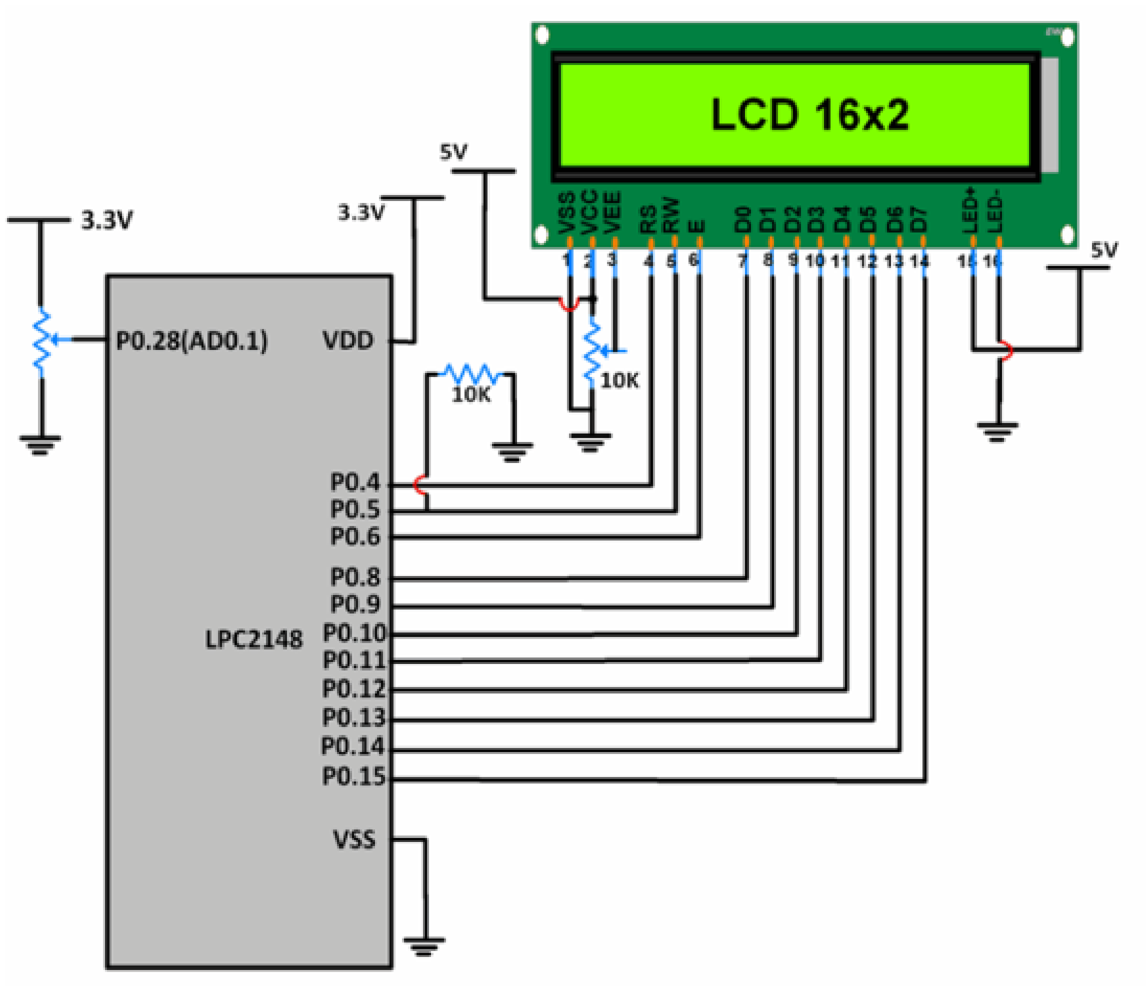
RAMAIAH Institute of Technology

Program (3)

//SQUARE

```
#include "lpc214x.h"
#include "stdint.h"


voiddelay_ms(uint16_t j)
{
uint16_tx, i;
        for(i=0;i<j;i++)
        {
for(x=0; x<6000; x++); |  /* loop to generate delay */
        }
}

int main (void)
{
uint16_t value;
uint16_ti = 0;

        PINSEL1 = 0x00080000;        /* P0.25 as DAC output */
        IO0DIR = 0xFFFFFFFF;
        while(1)
{
DACR=1023<<6;
delay_ms(10);
DACR=0;
delay_ms(10);

        }
}
```

ADC INTERFACE



Here, input signal is a DC signal which varies from 0 to 3.3V via a potentiometer.

Signal is given on P0.28. P0.28 is configured as AD0.1 using PINSEL register.

10-bit ADC result is stored in a variable and its lower 8 bits are given to P0.8-P0.15. These pins are connected to the data pins of an LCD.

P0.4,5,6 are used as RS, RW, EN pins of the LCD.

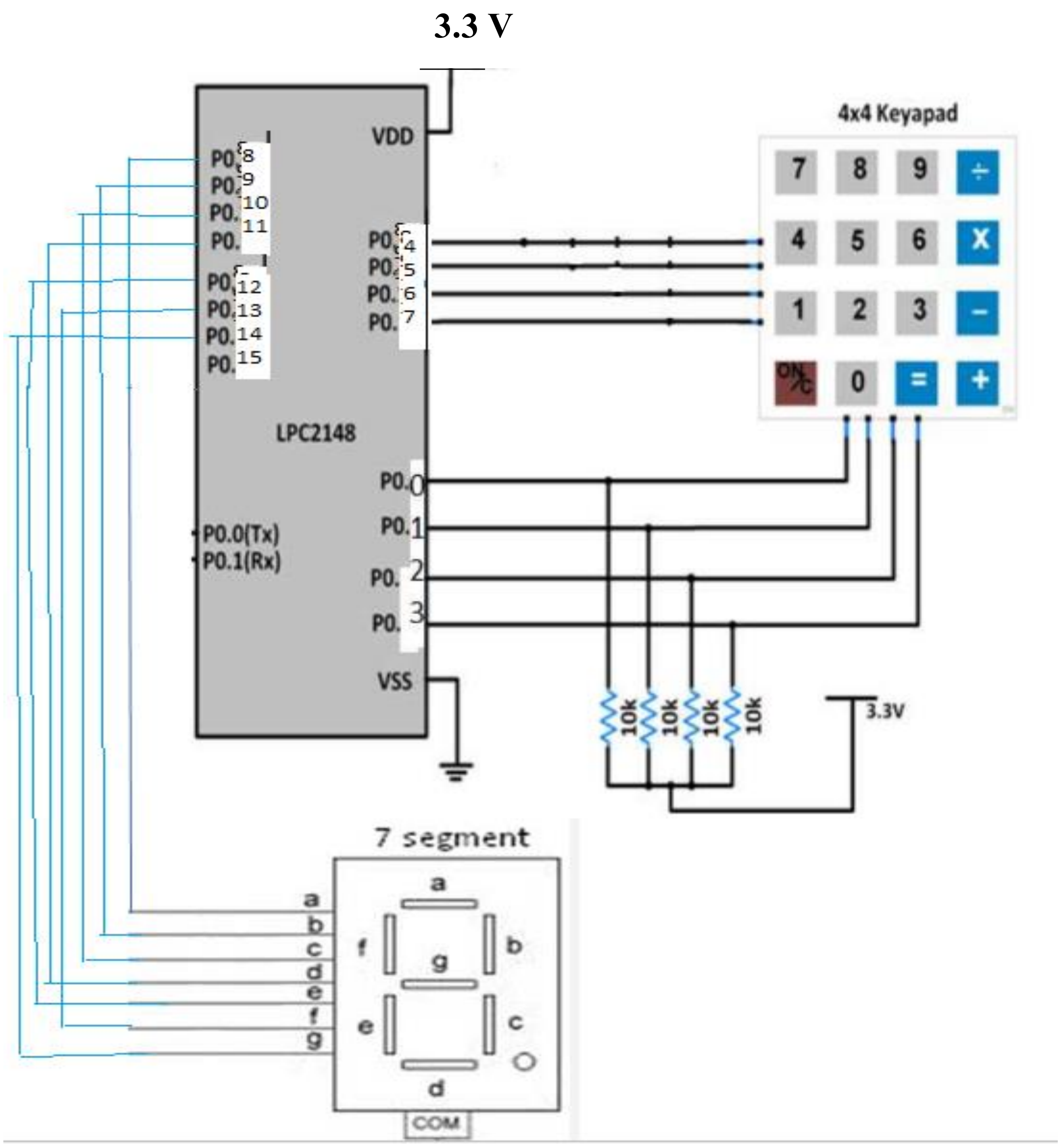
As the potentiometer is varied, we can see the variation in equivalent voltage value on the LCD.

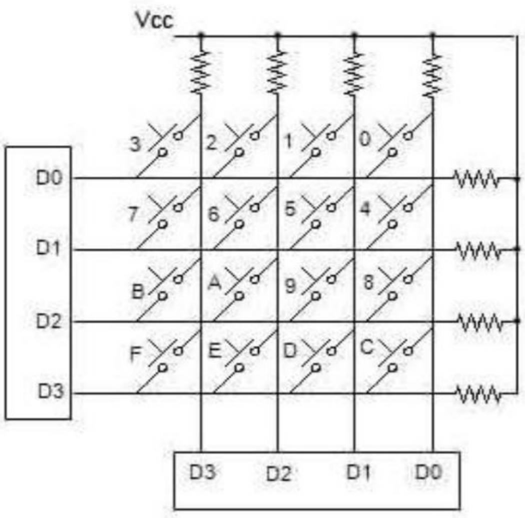
Write a Program using internal ADC in LPC2148 and display the variable voltage on LCD display

```
#include "lpc2148.h"
#include "stdint.h"
#include "stdio.h"
#include "string.h"

void InitLPC(void)
{

PINSEL0 = 0x00000000;
PINSEL1=0x01000000;
IO0DIR = 0XFFFFFFFF;
}

void delay_ms(uint16_t  j)                    /* Function for delay in milliseconds  */
{
   uint16_t x,i;
for(i=0;i<j;i++)
{
   for(x=0; x<6000; x++);
}
}

void LCD_Command(char command)
{
IO0SET = command<<8;
IO0SET = IO0SET |=0x00000040;                    /* EN = 1 */
delay_ms(2);
       IO0CLR = 0x00000040;
        IO0CLR=0xFFFFFFFF;
}

void LCD_Init(void)
{

LCD_Command(0x38);                                 /* Initialize lcd */
LCD_Command(0x0C);                                 /* Display on cursor off */
LCD_Command(0x06);                                /* Auto increment cursor */
LCD_Command(0x01);                                /* Display clear */

}

void LCD_String (char* msg)
{  uint8_t i=0;
   while(msg[i]!=0)
   {
IO0SET = msg[i]<<8;
IO0SET = IO0SET |=0x00000050;                    /* EN = 1 */
delay_ms(2);
       IO0CLR = 0x00000040;
       delay_ms(5);
       IO0CLR=0xFFFFFFFF;
       i++;
}

}
```

```
int main(void)
{
uint32_t result;
float voltage;
char adc[18];
 InitLPC();
 LCD_Init();

 AD0CR = 0x00200402;                    /* ADC operational, 10-bits, 11 clocks for conversion */
 while(1)
 {
 AD0CR = AD0CR | (1<<24);               /* Start Conversion */
 while ( !(AD0GDR & 0x80000000) );      /* Wait till DONE */
 result = AD0GDR;
 result = (result>>6);
 result = (result & 0x000003FF);
 voltage=result;
 LCD_Command(0x80);
 sprintf(adc, "DIG=%f V  ", voltage);
 LCD_String(adc);

 }
}
```

4 x 4 Keypad interface with LPC2148, and display the key pressed on a Seven segment LED display

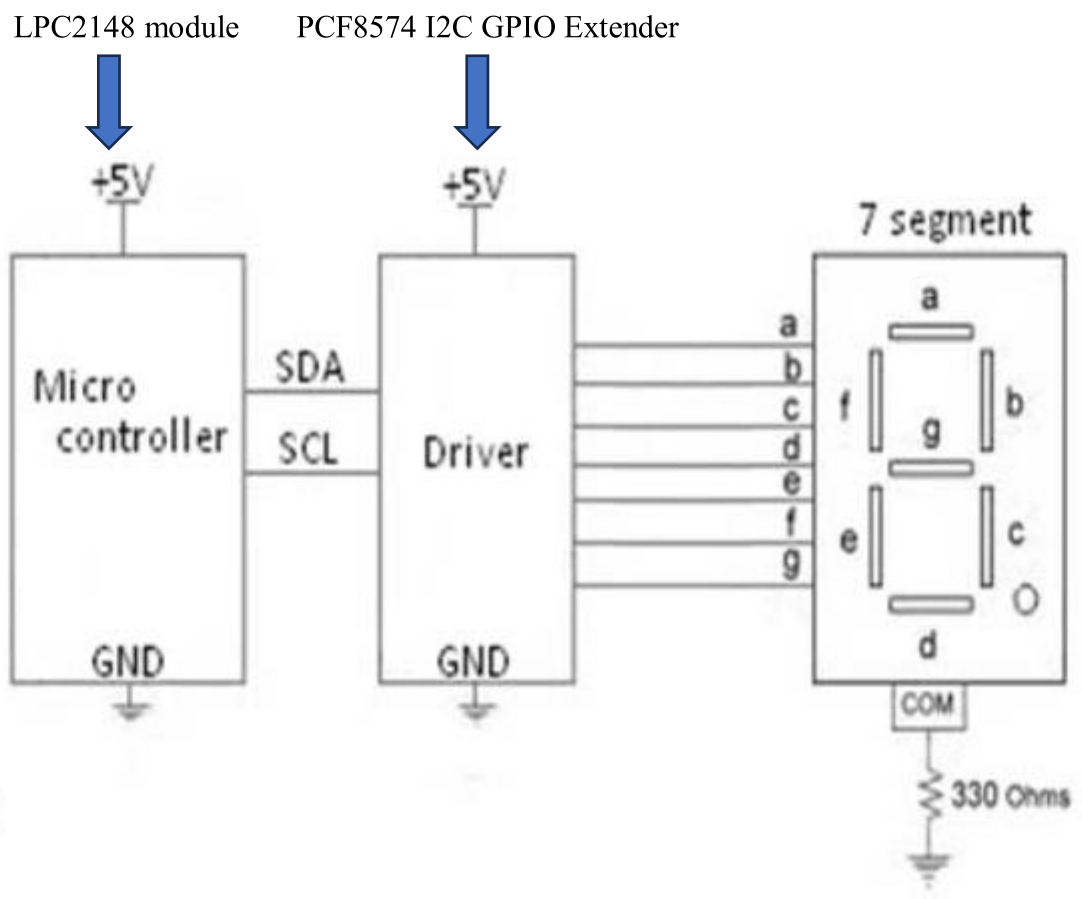


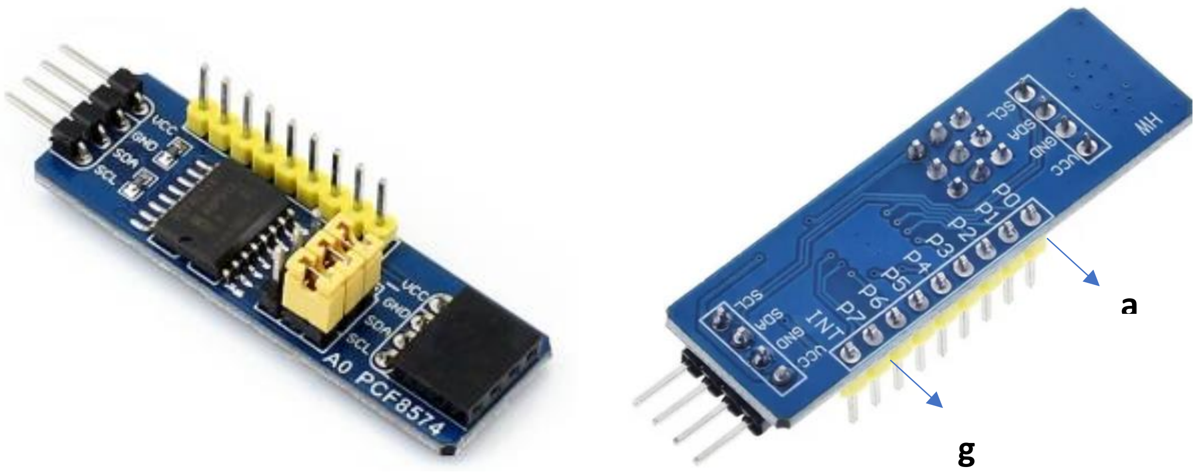


Interface keypad and display the key pressed on 7 segment LED display.

```
#include "lpc2148.h"
#include "stdint.h"
unsigned int i, delay_ms, segval;
unsigned char index, lcdval, row, keyscan, keyret, keynum=0, keypress, scanret = 0xFF;
unsigned char seg7[] = {0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x67, 0x77,
0x7c, 0x39, 0x5e, 0x79, 0x71, 0x00, 0x00, 0x0};
unsigned char scan[] = {0xEF,0xDF,0xBF,0x7F,0x00};
unsigned char keycode[] = {0xEE, 0xED, 0xEB, 0xE7, 0xDE, 0xDD, 0xDB, 0xD7, 0xBE,
0xBD, 0xBB, 0xB7, 0x7E, 0x7D, 0x7B, 0x77, 0x00};
Void InitLPC(void)
{
        PINSEL0 = 0x00L;
        IO0DIR =
        0XFFFFFFF0;
}
void Delay(unsigned int dms)
{       delay_ms = dms;
        while(delay_ms> 0)
                delay_ms--;
}
Void GetKey()
{
     row=0;
     while(1)
     {
          IO0CLR = 0xFF;
          row&= 0x3;
          keyscan=scan[row
          ]; IO0SET =
          keyscan;
          Delay(2);
          keyret = IO0PIN;
         if (keyscan !=
            keyret) break;
          row++;
     }
for(i=0;i<0x10;i++)
 {
   if(keycode[i]==keyret)
   keynum=i;
  }
 IO0CLR = 0xFF00;
 segval = seg7[keynum];
 segval<<= 8;
 IO0SET = segval;
 }

void main(void)
{
    InitLPC();
    index=0;
    while(1)
    GetKey();
}
```

Interfacing a Seven segment display to LPC2148 using I2C interface





PCF8574 I2C GPIO Extender

SCL	$\rightarrow$	P0.2 (LPC2148 Microcontroller)
SDA	$\rightarrow$	P0.3 (LPC2148 Microcontroller)
P0	$\rightarrow$	a (7 segment module)
P1	$\rightarrow$	b (7 segment module)
|		
P7	$\rightarrow$	h (7 segment module)

RAMAIAH Institute of Technology

Program to Interface Seven segment display to LPC2148 using I2C interface

```
#include "lpc2148.h"
#include <stdint.h>

void delay(unsigned int c)
{
unsigned int a;
for(a=0;a<=10000;a++);
}

#define I2EN (1<<6) //Enable/Disable bit
#define STA  (1<<5) //Start Set/Clear bit
#define STO  (1<<4) //Stop bit
#define SI   (1<<3) //Serial Interrupt Flag Clear bit
#define AA   (1<<2) //Assert Acknowledge Set/Clear bit

void waitforsi (void)
{
while (!(I2C0CONSET & SI));  //while interrupt in not set /wait till SI goes 1
}

void i2c_send_start (void)
{
I2C0CONSET = STA;  // set start bit
waitforsi();  //wait for interrupt to set it will transmit previous condition
}

void i2c_write (unsigned char data)
{
I2C0DAT = data;  // load data
I2C0CONCLR = SI|STA|AA;  //clear interrupt, start, ack
waitforsi ();  // wait for condition  to transmit
}

void i2c_send_stop (void)
{
I2C0CONCLR = AA|SI;  // clear ack and interrupt
I2C0CONSET = STO;  // send stop
}
```

```
void i2c_init (void)
{

I2C0SCLH = 75;  // 15MHz/150 = 100000 speed i2c
I2C0SCLL = 75;
I2C0CONCLR = STA|STO|AA|SI;  // clear all bits
I2C0CONSET = I2EN;  // enable bit set
}

void lcd_write_data (int data)
{
 unsigned char address = 0x40;
 unsigned char a[]={0x3f,0x6,0x5b,0x4f,0x66,0x6d,0x7d,0x7,0x7f,0x6f};
i2c_send_start();
i2c_write (address);
i2c_write (a[data]);
i2c_send_stop();
}

int main()
{
  int l;
  PINSEL0 = (1<<4)|(1<<6);  // for i2c line
  i2c_init ();
  while(1){
  for(l=0;l<10;l++){
  lcd_write_data(l);
  delay(2);
}
  }
}
```

Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of ECE
Program – UG - 2024

Term: 1st March – 22nd June 2024 **Sem:** VI **Sec:** A, B and C
Course: Embedded System Design Lab **Course Code:** ECL66

1. Write a C program to demonstrate usage of fork with child and parent process ID printed and parent waits for child process to terminate.

2. Write a C program using signal function calls as follows: when CTRL C is pressed a Signal, with message, "Press CTRL Z to terminate" should be sent.

3. Write a C program using signal function calls as follows: when CTRL C is pressed the signal should be ignored.

4. Write a C Program for creating Multithreading – One thread reads the input from the keyboard and another thread converts to upper case. This is done until "Stop" is pressed. Number of threads can be running sharing same CPU.

5. Write a C Program for creating two threads, one for reading the input and one for converting the text to upper case letters. This is done until "Stop" is pressed. Converting thread will wait for a semaphore to be released before it starts the operation.

6. Write a C program using pthreads that demonstrates the use of a mutex to synchronize access to a shared variable by two threads. One thread should increment the variable, while the other thread should decrement it.

7. Write a program for IPC using Message Queues to send data to a message queue and write another program for IPC using Message Queues to receive or read message from the above created message queue.

8. Write an embedded C Program to read an analog input using ADC module of LPC2148, and display the variable voltage on LCD display.

9. Interface DAC module to LPC2148 microcontroller and write an embedded C code to generate a sinusoidal waveform and display on CRO.

10. Interface DAC module to LPC2148 microcontroller and write an embedded C code to generate a triangular waveform and display on CRO

11. Interface DAC module to LPC2148 microcontroller and write an embedded C code to generate a square waveform and display on CRO.

12. Interface a 4x4 keypad to LPC2148 microcontroller, and write an embedded C code to display the key pressed on the Seven Segment LED display.

13. Interface one, Seven-Segment display module to LPC2148 microcontroller using I2C interface for serial communication and write an embedded C code for a BCD counter to count from 0 - 9

Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of ---ECE
Program - UG

Term: 1st March – 22nd June 2024 **Sem:** VI **Sec:** A, B and C
Course: Embedded System Design Lab **Course Code:** ECL66

Scheme for Embedded System Design Laboratory

Sl.No.			**50 Marks**
1.	Two programs : Write Up	16% of 50 marks	8
2.	Two programs : Execution + Result	70% of 50 marks	35
3.	Viva	14% of 50 marks	7

Note : Change of experiment should be taken within 30 minutes from the commencement of exam 20% marks is deduced from the total marks, if a change of experiment is taken.