**PROGRAM No. 1**

**Read two strings, store them in locations STR1 and STR2. Check whether they are equal or not and display appropriated messages. Also display the length of the stored strings.**

disp macro msg

lea dx,msg

mov ah,9

int 21h

endm

.model small

.stack

.data

m1 db 10,13,"enter string 1:$"

m2 db 10,13,"enter string 2:$"

m3 db 10,13,"length of string 1 is:$"

m4 db 10,13,"length of string 2 is:$"

m5 db 10,13,"string1 equal to string2$"

m6 db 10,13,"string1 not equal to string2$"

str1 db 80 dup(40)

str2 db 80 dup(40)

l1 db ?

l2 db ?

.code

mov ax,@data

mov ds,ax

mov es,ax

disp m1

lea dx,str1

call read

disp m2

lea dx,str2

call read

mov al,[str1+1]

mov l1,al

mov al,[str2+1]

mov l2,al

cmp al,l1

jne strnote

mov ch,0

mov cl,l1

lea si, str1+2

lea di, str2+2

cld

repe cmpsb

jne strnote

disp m5

jmp next

strnote: disp m6

next: disp m3

mov al, l1

call displ

disp m4

mov al, l2

call displ

mov ah,4ch

int 21h

read proc

mov ah,0ah

int 21h

ret

read endp

displ proc

aam

mov bx,ax

add bx,3030h

mov ah,2

mov dl,bh

int 21h

mov dl,bl

int 21h

ret

displ endp

end

**PROGRAM No. 2**

**Simulate a Decimal Up-counter to display 00-99.**

.model small

.stack

.data

msg db "press any key to exit$"

.code

mov ax,@data

mov ds,ax

call clear

lea dx,msg

mov ah,9

int 21h

mov ax,00h

nxtnum:push ax

call setcursor

call disp

call delay

mov ah,01h

int 16h

jnz exit

pop ax

add ax,1

daa

cmp ax,0

jnz nxtnum

exit: mov ah,4ch

int 21h

setcursor proc

mov ah,2

mov dh,12

mov dl,40

int 10h

ret

setcursor endp

disp proc

mov bl,al

mov dl,al

mov cl,4

shr dl,cl

add dl,30h

mov ah,2

int 21h

mov dl,bl

and dl,0fh

add dl,30h

int 21h

ret

disp endp

delay proc

mov bx,00ffh

b2:mov cx,0ffffh

b1:loop b1

dec bx

jnz b2

ret

delay endp

clear proc

mov al,0

mov ah,6

mov ch,0

mov cl,0

mov dh,24

mov dl,79

mov bh,7

int 10h

ret

clear endp

end

**PROGRAM No. 3**

**Compute nCr using recursive procedure. Assume that ‘n’ and ‘r’ are non- negative integers.**

.model small

.stack

.data

n dw 4

r dw 2

ncr dw 0

msg db "ncr= $"

.code

mov ax,@data

mov ds,ax

mov ax,n

mov bx,r

call ncrpro

mov ax,ncr

mov bx,ax

lea dx,msg

mov ah,9

int 21h

mov ax,bx

aam

mov bx,ax

add bx,3030h

mov dl,bh

mov ah,2

int 21h

mov dl,bl

int 21h

mov ah,4ch

int 21h

ncrpro proc near

cmp bx,ax

je res1

cmp bx,0

je res1

cmp bx,1

je resn

dec ax

cmp bx,ax

je incr

push ax

push bx

call ncrpro

pop bx

pop ax

dec bx

push ax

push bx

call ncrpro

pop bx

pop ax

ret

res1:inc ncr

ret

incr:inc ncr

resn:add ncr,ax

ret

ncrpro endp

end

**PROGRAM No. 4**

**Sort a given set of ‘n’ numbers in ascending and descending orders using the Bubble Sort algorithm.**

.model small

.stack 100

.data

a db 10,6,8,0,4,2

len dw($-a)

.code

start: mov ax,@data

mov ds,ax

mov bx,len

dec bx

outloop:mov cx,bx

mov si,0

inloop: mov al, a[si]

cmp al,a[si+1]

jb next

xchg al,a[si+1]

mov a[si],al

next: inc si

loop inloop

dec bx

jnz outloop

mov ah, 4ch

int 21h

end start

**PROGRAM No. 5**

**Read the current time from the system and display it in the standard format on the screen.**

.model small

.stack

.data

msg db 10,13,"current time is $"

.code

mov ax,@data

mov ds,ax

lea dx,msg

mov ah,9

int 21h

mov ah,2ch

int 21h

mov al,ch

call disp

mov dl,':'

mov ah,2

int 21h

mov al,cl

call disp

mov dl,':'

mov ah,2

int 21h

mov al,dh

call disp

mov dl,'.'

mov ah,2

int 21h

mov ah,4ch

int 21h

disp proc near

aam

add ax,3030h

mov bx,ax

mov dl,bh

mov ah,2

int 21h

mov dl,bl

int 21h

ret

disp endp

end

**PROGRAM No. 6**

**i) Interface a Stepper motor and rotate it in clockwise and anti-clockwise direction.**

#include <LPC21xx.H>

void clock\_wise(void);

void anti\_clock\_wise(void);

unsigned long int var1, var2;

unsigned int i=0, j=0, k=0;

int main(void)

{

PINSEL0 = 0x00FFFFFF; //P0.12 to P0.15 GPIO

IO0DIR |= 0x0000F000; //P0.12 to P0.15 output

while(1)

{

for(j=0;j<50;j++) // 20 times in Clock wise Rotation

clock\_wise();

for(k=0;k<65000;k++); // Delay to show anti\_clock Rotation

for(j=0;j<50;j++) // 20 times in Anti Clock wise Rotation

anti\_clock\_wise();

for(k=0;k<65000;k++); // Delay to show clock Rotation

} // End of while(1)

} // End of main

void clock\_wise(void)

{

var1 = 0x00000800; //For Clockwise

for(i=0;i<=3;i++) // for A B C D Stepping

{

var1 = var1<<1; //For Clockwise

var2 = ~var1;

var2 = var2 & 0x0000F000;

IO0PIN = ~var2;

for(k=0;k<3000;k++); //for step speed variation

}

}

void anti\_clock\_wise(void)

{

var1 = 0x00010000; //For Anticlockwise

for(i=0;i<=3;i++) // for A B C D Stepping

{

var1 = var1>>1; //For Anticlockwise

var2 = ~var1;

var2 = var2 & 0x0000F000;

IO0PIN = ~var2;

for(k=0;k<3000;k++); //for step speed variation

}

}

**ii) Interface and Control a DC Motor.**

#include<lpc214x.h>

void clock\_wise(void);

void anti\_clock\_wise(void);

unsigned int j=0;

int main()

{

IO0DIR= 0X00000900;

IO0SET= 0X00000100; //P0.8 should always high.

while(1)

{

clock\_wise();

for(j=0;j<400000;j++); //delay

anti\_clock\_wise();

for(j=0;j<400000;j++); //delay

} //End of while(1)

} //End of Main

void clock\_wise(void)

{

IO0CLR = 0x00000900; //stop motor and also turn off relay

for(j=0;j<10000;j++); //small delay to allow motor to turn off

IO0SET = 0X00000900; //Selecting the P0.11 line for clockwise and turn on motor

}

void anti\_clock\_wise(void)

{

IO0CLR = 0X00000900; //stop motor and also turn off relay

for(j=0;j<10000;j++); //small delay to allow motor to turn off

IO0SET = 0X00000100; //not selecting the P0.11 line for Anti clockwise

}

**PROGRAM No. 7**

**Display “Hello World” message using Internal UART.**

#include <lpc214x.h>

#include<stdio.h>

const char \*msg="HELLO WORLD\r",\*ptr;

int main(void)

{

PINSEL0=0X0000005; //P0.0,P0.1-select TXD0 and RXD0 lines

U0LCR = 0X00000083; //DLAB=1, 1 STOP BIT,8-BIT CHARACTER LENGTH

U0DLM = 0X00; //select the data format

U0DLL = 0x13; //select baud rate 9600 bps from formula Pclk=3MHZ

U0LCR = 0X00000003; //DLAB=0

ptr=msg; //for continuous printing

while(1)

{

while (\*msg!=0x00)

{

while(!(U0LSR & 0X20));

U0THR = \*msg;

msg++;

}

msg=ptr; //for printing continuously

//printf("HELLO WORLD!");

}

}

**PROGRAM No. 8**

**Display the 4-digit counter sequence 000, 001, …. FFF on a 7-segment LED interface, with an appropriate delay in between.**

#include <LPC21XX.h>

unsigned int delay;

unsigned int Switchcount=0;

unsigned int Disp[16]={0x003F0000, 0x00060000, 0x005B0000, 0x004F0000, 0x00660000,0x006D0000,

0x007D0000, 0x00070000, 0x007F0000, 0x006F0000,

0x00770000,0x007C0000, 0x00390000, 0x005E0000, 0x00790000,

0x00710000 };

#define ALLDISP 0xF0000000 //Select all display

#define DATAPORT 0x00FF0000 //P0.16 to P0.23 Data lines connected to drive Seven Segments

int main (void)

{

PINSEL1 = 0x00000000;

IO0DIR = 0xF0FF0000;

while(1)

{

IO0SET = ALLDISP; // select all digits

IO0CLR = 0x00FF0000; // clear the data lines to 7-segment displays

IO0SET = Disp[Switchcount]; // get the 7-segment display value from the array

for(delay=0; delay<500000;delay++) // delay

{}

Switchcount++;

if(Switchcount == 0x10) // 0 to F has been displayed ? go back to 0

{

Switchcount = 0;

IO0CLR = 0xF0FF0000;

}

}

}

**PROGRAM No. 9**

**Determine Digital output for a given Analog input using Internal ADC of ARM controller.**

#include <lpc214x.h>

#include <Stdio.h>

#include "lcd\_h.h"

unsigned int adc\_value=0,temp\_adc=0;

float adc\_ip;

char var[15],var1[15];

char \*ptr,arr[]= "ADC O/P= ";

char \*ptr1,dis[]="A I/P = ";

#define vol 3.3 //Reference voltage

#define fullscale 0x3ff //10 bit adc

int main()

{

PINSEL1 = 0X00040000; //AD0.4 pin is selected(P0.25)

IO0DIR = 0x000000FC; //configure o/p lines for lcd

lcd\_init(); //LCD initialization

delay(3200);

ptr = dis;

temp1 = 0x80; //Display starting address of first line 1 th pos

lcd\_com();

delay(800);

while(\*ptr!='\0')

{

temp1 = \*ptr;

lcd\_data();

ptr ++;

}

ptr1 = arr;

temp1 = 0xC0; //Display starting address of second line 4 th pos

lcd\_com();

delay(800);

while(\*ptr1!='\0')

{

temp1 = \*ptr1;

lcd\_data();

ptr1 ++;

}

//infinite loop

while(1)

{

//CONTROL register for ADC

AD0CR = 0x01200010; //command register for ADC-AD0.4

while(((temp\_adc = AD0GDR) &0x80000000) == 0x00000000); //to check the interrupt bit

adc\_value = AD0GDR; //reading the ADC value

adc\_value >>=6;

adc\_value &= 0x000003ff;

adc\_ip = ((float)adc\_value \* (float)vol)/(float)fullscale;

sprintf(var1,"%4.2fV",adc\_ip);

sprintf(var,"%3x",adc\_value);

temp1 = 0x89;

lcd\_com();

delay(1200);

ptr = var1;

while(\*ptr!='\0')

{

temp1=\*ptr;

lcd\_data();

ptr++;

}

temp1 = 0xc9;

lcd\_com();

delay(1200);

ptr1 = var;

while(\*ptr1!='\0')

{

temp1=\*ptr1;

lcd\_data();

ptr1++;

}

} // end of while(1)

} //end of main()

**PROGRAM No. 10**

**Interface a 4x4 keyboard and display the key code on an LCD.**

#include<lpc21xx.h>

#include<stdio.h>

/\*\*\*\*\*\*\* FUNCTION PROTOTYPE\*\*\*\*\*\*\*/

void lcd\_init(void);

void clr\_disp(void);

void lcd\_com(void);

void lcd\_data(void);

void wr\_cn(void);

void wr\_dn(void);

void scan(void);

void get\_key(void);

void display(void);

void delay(unsigned int);

void init\_port(void);

unsigned long int scan\_code[16]= {0x00EE0000,0x00ED0000,0x00EB0000,0x00E70000,

0x00DE0000,0x00DD0000,0x00DB0000,0x00D70000,

0x00BE0000,0x00BD0000,0x00BB0000,0x00B70000,

0x007E0000,0x007D0000,0x007B0000,0x00770000};

unsigned char ASCII\_CODE[16]= {'0','1','2','3',

'4','5','6','7',

'8','9','A','B',

'C','D','E','F'};

unsigned char row,col;

unsigned char temp,flag,i,result,temp1;

unsigned int r,r1;

unsigned long int var,var1,var2,res1,temp2,temp3,temp4;

unsigned char \*ptr,disp[] = "4X4 KEYPAD";

unsigned char disp0[] = "KEYPAD TESTING";

unsigned char disp1[] = "KEY = ";

int main()

{

// \_\_ARMLIB\_enableIRQ();

init\_port(); //port intialisation

delay(3200); //delay

lcd\_init(); //lcd intialisation

delay(3200); //delay

clr\_disp(); //clear display

delay(500); //delay

//........LCD DISPLAY TEST.........//

ptr = disp;

temp1 = 0x81; // Display starting address

lcd\_com();

delay(800);

while(\*ptr!='\0')

{

temp1 = \*ptr;

lcd\_data();

ptr ++;

}

//........KEYPAD Working.........//

while(1)

{

get\_key();

display();

}

} //end of main()

void get\_key(void) //get the key from the keyboard

{

unsigned int i;

flag = 0x00;

IO1PIN=0x000f0000;

while(1)

{

for(row=0X00;row<0X04;row++) //Writing one for col's

{

if( row == 0X00)

{

temp3=0x00700000;

}

else if(row == 0X01)

{

temp3=0x00B00000;

}

else if(row == 0X02)

{

temp3=0x00D00000;

}

else if(row == 0X03)

{

temp3=0x00E00000;

}

var1 = temp3;

IO1PIN = var1; // each time var1 value is put to port1

IO1CLR =~var1; // Once again Conforming (clearing all other bits)

scan();

delay(100); //delay

if(flag == 0xff)

break;

} // end of for

if(flag == 0xff)

break;

} // end of while

for(i=0;i<16;i++)

{

if(scan\_code[i] == res1) //equate the scan\_code with res1

{

result = ASCII\_CODE[i]; //same position value of ascii code

break; //is assigned to result

}

}

}// end of get\_key();

void scan(void)

{

unsigned long int t;

temp2 = IO1PIN; // status of port1

temp2 = temp2 & 0x000F0000; // Verifying column key

if(temp2 != 0x000F0000) // Check for Key Press or Not

{

delay(1000); //delay(100)//give debounce delay check again

temp2 = IO1PIN;

temp2 = temp2 & 0x000F0000; //changed condition is same

if(temp2 != 0x000F0000) // store the value in res1

{

flag = 0xff;

res1 = temp2;

t = (temp3 & 0x00F00000); //Verfying Row Write

res1 = res1 | t; //final scan value is stored in res1

}

else

{

flag = 0x00;

}

}

} // end of scan()

void display(void)

{

ptr = disp0;

temp1 = 0x80; // Display starting address of first line

lcd\_com();

while(\*ptr!='\0')

{

temp1 = \*ptr;

lcd\_data();

ptr ++;

}

ptr = disp1;

temp1 = 0xC0; // Display starting address of second line

lcd\_com();

while(\*ptr!='\0')

{

temp1 = \*ptr;

lcd\_data();

ptr ++;

}

temp1 = 0xC6; //display address for key value

lcd\_com();

temp1 = result;

lcd\_data();

}

void lcd\_init (void)

{

temp = 0x30;

wr\_cn();

delay(3200);

temp = 0x30;

wr\_cn();

delay(3200);

temp = 0x30;

wr\_cn();

delay(3200);

temp = 0x20;

wr\_cn();

delay(3200);

// load command for lcd function setting with lcd in 4 bit mode,

// 2 line and 5x7 matrix display

temp = 0x28;

lcd\_com();

delay(3200);

// load a command for display on, cursor on and blinking off

temp1 = 0x0C;

lcd\_com();

delay(800);

// command for cursor increment after data dump

temp1 = 0x06;

lcd\_com();

delay(800);

temp1 = 0x80;

lcd\_com();

delay(800);

}

void lcd\_data(void)

{

temp = temp1 & 0xf0;

wr\_dn();

temp= temp1 & 0x0f;

temp= temp << 4;

wr\_dn();

delay(100);

}

void wr\_dn(void) ////write data reg

{

IO0CLR = 0x000000FC; // clear the port lines.

IO0SET = temp; // Assign the value to the PORT lines

IO0SET = 0x00000004; // set bit RS = 1

IO0SET = 0x00000008; // E=1

delay(10);

IO0CLR = 0x00000008;

}

void lcd\_com(void)

{

temp = temp1 & 0xf0;

wr\_cn();

temp = temp1 & 0x0f;

temp = temp << 4;

wr\_cn();

delay(500);

}

void wr\_cn(void) //write command reg

{

IO0CLR = 0x000000FC; // clear the port lines.

IO0SET = temp; // Assign the value to the PORT lines

IO0CLR = 0x00000004; // clear bit RS = 0

IO0SET = 0x00000008; // E=1

delay(10);

IO0CLR = 0x00000008;

}

void clr\_disp(void)

{

// command to clear lcd display

temp1 = 0x01;

lcd\_com();

delay(500);

}

void delay(unsigned int r1)

{

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

}

void init\_port()

{

IO0DIR = 0x000000FC; //configure o/p lines for lcd

IO1DIR = 0XFFF0FFFF;

}