1.SEGMENT LED BLINKING

C-Program for Interfacing LED to LPC1768 and operating it at various delays:

```
#include <lpc17xx.h>
void delay_ms(unsigned int ms)
{
  unsigned int i,j;
  for(i=0;i<ms;i++)
    for(j=0;j<20000;j++);
}
/* start the main program */
int main()
{
GPIO;
LPC_GPIO2->FIODIR = 0xffffffff; //Configure the PORT2 pins as
 //Clock and PLL configuration LPC_PINCON->PINSEL4 = 0x000000; //Configure the
PORT2 Pins as
SystemInit();
OUTPUT;
while(1)
   {
    //PORT2  P2.0.to.P2.7 data  P2.8 sel
    LPC_GPIO2->FIOSET = 0xffffE40;                // '0'--->0x40
         delay_ms(1000);
         LPC_GPIO2->FIOSET = 0xfffffe79; //   '1'--->0x79
         delay_ms(1000);
         LPC_GPIO2->FIOSET = 0xfffffe24; //   '2'--->0x24
         delay_ms(1000);
         LPC_GPIO2->FIOSET = 0xfffffe30;//   '3'--->0x30

         delay_ms(1000);
LPC_GPIO2->FIOSET = 0xfffffe19;//   '4'--->0x19
delay_ms(1000);
LPC_GPIO2->FIOSET = 0xfffffe12;//   '5'--->0x12
delay_ms(1000);
LPC_GPIO2->FIOSET = 0xfffffe02;//   '6'--->0x02
delay_ms(1000);
 LPC_GPIO2->FIOSET = 0xfffffe78; //   '7'--->0x78
delay_ms(1000);
LPC_GPIO2->FIOSET = 0xfffffe00;//   '8'--->0x00
delay_ms(1000);
LPC_GPIO2->FIOSET = 0xfffffe18;//   '9'--->0x18
delay_ms(1000);
LPC_GPIO2->FIOSET = 0xfffffe08;//   'A'--->0x08
delay_ms(1000);
LPC_GPIO2->FIOSET = 0xfffffe00;//   'B'--->0x00
delay_ms(1000);
LPC_GPIO2->FIOSET = 0xfffffe46;//   'C'--->0x46
```

```
delay_ms(1000);
 LPC_GPIO2->FIOSET = 0xfffffe40; //   'D'--->0x40
delay_ms(1000);
LPC_GPIO2->FIOSET = 0xfffffe06;//   'E'--->0x06
delay_ms(1000);
LPC_GPIO2->FIOSET = 0xfffffe0e;//   'F'--->0x0E
delay_ms(1000);
}
}
```

==========

2. KEYPAD AND LCD DISPLAY

C-Program for Interfacing Keypad and LCD Display to LPC1768:

```
#include "keypad.h"
#include "delay.h"
#include "gpio.h"
gpioPins_et A_RowsPins_U8[C_MaxRows_U8]; gpioPins_et
A_ColsPins_U8[C_MaxCols_U8];
const uint8_t A_KeyLookUptable_U8[C_MaxRows_U8][C_MaxCols_U8]=
{
    '0','1','2','3',
'4','5','6','7',
'8','9','A','B',
'C','D','E','F'
};
/*********************************************************************** ****************************
local function prototypes ***********************************************************************
****************************/
static void keypad_WaitForKeyRelease(void);
static void keypad_WaitForKeyPress(void);
/*********************************************************************** ****************************/
/*********************************************************************** ****************************
                void KEYPAD_Init()
*********************************************************************** ****************************
* I/P Arguments:Pin numbers where the rows and columns are conneted. * Return
value : none
  '0','1','2','3',
'4','5','6','7',
'8','9','A','B',
'C','D','E','F'

* description : This function configures the rows and columns for keypad scan
      1.ROW lines are configured as Output.
      2.Column Lines are configured as Input.
*********************************************************************** ****************************/
void KEYPAD_Init(
      gpioPins_et row_0,
```

```
        gpioPins_et row_1,
        gpioPins_et row_2,
        gpioPins_et row_3,
        gpioPins_et col_0,
        gpioPins_et col_1,
        gpioPins_et col_2,
        gpioPins_et col_3 )
{
uint8_t i;
   A_RowsPins_U8[0] = row_0;
   A_RowsPins_U8[1] = row_1;
   A_RowsPins_U8[2] = row_2;
   A_RowsPins_U8[3] = row_3;
   A_ColsPins_U8[0] = col_0;
   A_ColsPins_U8[1] = col_1;
   A_ColsPins_U8[2] = col_2;
   A_ColsPins_U8[3] = col_3;
   for(i=0;i<C_MaxRows_U8;i++)
   {
GPIO_PinDirection(A_RowsPins_U8[i],OUTPUT); }
   for(i=0;i<C_MaxCols_U8;i++)
   {
GPIO_PinDirection(A_ColsPins_U8[i],INPUT); }
} /********************************************************************** ******************************
static void keypad_WaitForKeyRelease(void)
**********************************************************************
*****************************
 * I/P Arguments:none
* Return value : none
* description : This function waits till the previous key is released.
********************************************************************** ****************************/
static void keypad_WaitForKeyRelease(void) {
   uint8_t i,v_keyStatus_u8;
   for (i=0;i<C_MaxRows_U8;i++)
{
GPIO_PinWrite(A_RowsPins_U8[i],LOW);
}

do
{
do {
v_keyStatus_u8 = 1;
for(i=0; i<C_MaxCols_U8; i++)
{
v_keyStatus_u8 &= GPIO_PinRead(A_ColsPins_U8[i]); }
}while(v_keyStatus_u8 == 0); DELAY_us(C_DebounceTimeInMicroSecond_U16);
       v_keyStatus_u8 = 1;
       for(i=0; i<C_MaxCols_U8; i++)
```

```
{
v_keyStatus_u8 &= GPIO_PinRead(A_ColsPins_U8[i]);
      }
   }while(v_keyStatus_u8 == 0);
} /********************************************************************** ****************************
static void keypad_WaitForKeyPress(void)
********************************************************************** ****************************
* I/P Arguments:none * Return value : none
* description : This function waits till a new key is pressed.
The new Key pressed can be decoded by the function KEYPAD_GetKey
********************************************************************** ****************************/
static void keypad_WaitForKeyPress(void)
{
   uint8_t i,v_keyStatus_u8;
   for (i=0;i<C_MaxRows_U8;i++)
{
GPIO_PinWrite(A_RowsPins_U8[i],LOW);
}
do {
do {
         }
      }while(v_keyStatus_u8 != 0);
v_keyStatus_u8 = 1;
for(i=0; i<C_MaxCols_U8; i++)
{
v_keyStatus_u8 &= GPIO_PinRead(A_ColsPins_U8[i]);

DELAY_us(C_DebounceTimeInMicroSecond_U16);
      v_keyStatus_u8 = 1;
      for(i=0; i<C_MaxCols_U8; i++)
{
v_keyStatus_u8 &= GPIO_PinRead(A_ColsPins_U8[i]);
      }
   }while(v_keyStatus_u8 != 0);
}
/********************************************************************** ****************************
            unsigned char KEYPAD_GetKey()
********************************************************************** ****************************
 * I/P Arguments:none
* Return value : uint8_t--> ASCII value of the Key Pressed
* description: This function waits till a key is pressed and returns
its ASCII Value
pressed:
key.
It follows the following sequences to decode the key
1.Wait till the previous key is released..
2.Wait for the new key press.
3.Scan all the rows one at a time for the pressed
```

4.Decodes the key pressed depending on ROW-COL combination and returns its ASCII value.

**********/

```
uint8_t KEYPAD_GetKey(void)
{
   uint8_t i,j,v_KeyPressed_u8 = 0;
   keypad_WaitForKeyRelease();
   keypad_WaitForKeyPress();
   for (i=0;i<C_MaxRows_U8;i++)
{
GPIO_PinWrite(A_RowsPins_U8[i],HIGH);
}
   for (i=0;(i<C_MaxRows_U8);i++)
{
GPIO_PinWrite(A_RowsPins_U8[i],LOW);
      for(j=0; (j<C_MaxCols_U8); j++)
{
if(GPIO_PinRead(A_ColsPins_U8[j]) == 0) {
            v_KeyPressed_u8 = 1;

break; }
}
      if(v_KeyPressed_u8 ==1)
{
break;
}
GPIO_PinWrite(A_RowsPins_U8[i],HIGH); }
if(i<C_MaxRows_U8)
v_KeyPressed_u8 = A_KeyLookUptable_U8[i][j];
   else
      v_KeyPressed_u8 = C_DefaultKey_U8;
   return v_KeyPressed_u8;
}
```

==========

3. STEPPER MOTOR

C-Program for Interfacing Stepper motor to LPC1768:

```
#include <lpc17xx.h>
void delay_ms(unsigned int ms)
{
  unsigned int i,j;
  for(i=0;i<ms;i++)
    for(j=0;j<20000;j++);
}
void Forword()
{
     LPC_GPIO2->FIOSET = 0xfffffffA;
    delay_ms(10);
```

```
    LPC_GPIO2->FIOSET = 0xfffffff6;
        delay_ms(10);
    LPC_GPIO2->FIOSET = 0xfffffff5;
        delay_ms(10);
    LPC_GPIO2->FIOSET = 0xfffffff9;
        delay_ms(10);
}
void Reverse()
 {
 LPC_GPIO2->FIOSET = 0xfffffff9;
delay_ms(10);
LPC_GPIO2->FIOSET = 0xfffffff5;
    delay_ms(10);
LPC_GPIO2->FIOSET = 0xfffffff6;
    delay_ms(10);
LPC_GPIO2->FIOSET = 0xfffffffA;

delay_ms(10);
}

/* start the main program */
int main()
{
SystemInit(); //Clock and PLL configuration
LPC_PINCON->PINSEL4 = 0x000000; //Configure the PORT2 Pins as GPIO;
LPC_GPIO2->FIODIR = 0xffffffff; //Configure the PORT2 pins as OUTPUT;
LPC_GPIO2->FIOSET = 0xfffffff0;
  while(1)
{ Forword();
} }
```

==========

4. UART INTERFACING

C-Program for Interfacing 7-Segment LED Display to LPC1768:

```
 #include <lpc17xx.h>
#include "stdutils.h"
#define SBIT_WordLenght 0x00u
#define SBIT_DLAB 0x07u
#define SBIT_FIFO 0x00u
#define SBIT_RxFIFO 0x01u
#define SBIT_TxFIFO 0x02u
#define SBIT_RDR 0x00u
#define SBIT_THRE 0x05u
/* Function to initialize the UART0 at specifief baud rate */ void uart_init(uint32_t
baudrate)
{
uint32_t var_UartPclk_u32,var_Pclk_u32,var_RegValue_u32;
LPC_PINCON->PINSEL0 &= ~0x000000F0;
```

```
LPC_PINCON->PINSEL0 |= 0x00000050; // Enable TxD0 P0.2 and p0.3
LPC_UART0->FCR = (1<<SBIT_FIFO) | (1<<SBIT_RxFIFO) | (1<<SBIT_TxFIFO); //
Enable FIFO and reset Rx/Tx FIFO buffers
LPC_UART0->LCR = (0x03<<SBIT_WordLenght) | (1<<SBIT_DLAB); // 8bit data, 1Stop
bit, No parity
/** Baud Rate Calculation :
PCLKSELx registers contains the PCLK info for all the clock
dependent peripherals.
Bit6,Bit7 contains the Uart Clock(ie.UART_PCLK) information. The UART_PCLK and
the actual Peripheral Clock(PCLK) is
calculated as below.
      (Refer data sheet for more info)

      UART_PCLK    PCLK
        0x00       SystemFreq/4
        0x01       SystemFreq
        0x02       SystemFreq/2
        0x03       SystemFreq/8
**/
var_UartPclk_u32 = (LPC_SC->PCLKSEL0 >> 6) & 0x03;
   switch( var_UartPclk_u32 )
   {
   case 0x00:
      var_Pclk_u32 = SystemCoreClock/4;
      break;
   case 0x01:
      var_Pclk_u32 = SystemCoreClock;
      break;
   case 0x02:
      var_Pclk_u32 = SystemCoreClock/2;
      break;
   case 0x03:
      var_Pclk_u32 = SystemCoreClock/8;
break; }
var_RegValue_u32 = ( var_Pclk_u32 / (16 * baudrate ));
 LPC_UART0->DLL = var_RegValue_u32 & 0xFF;
LPC_UART0->DLM = (var_RegValue_u32 >> 0x08) & 0xFF;
util_BitClear(LPC_UART0->LCR,(SBIT_DLAB)); // Clear DLAB after setting DLL,DLM
}
/* Function to transmit a char */
void uart_TxChar(char ch)
{
while(util_IsBitCleared(LPC_UART0->LSR,SBIT_THRE)); // Wait for
Previous transmission
   LPC_UART0->THR=ch;
data to be transmitted
}
/* Function to Receive a char */
```

```
char uart_RxChar()
{
char ch;
// Load the
while(util_IsBitCleared(LPC_UART0->LSR,SBIT_RDR)); // Wait till the data is received
ch = LPC_UART0->RBR; // Read received data
return ch; }

int main()
{
   char ch,a[]="RDL TECHNOLOGIES";
   int i;
SystemInit();
uart_init(9600); // Initialize the UART0 for 9600 l'll baud rate
   for(i=0;a[i];i++)  //transmit a predefined string
      uart_TxChar(a[i]);
while(1)
   {
      //Finally receive a char and transmit it infinitely
      ch = uart_RxChar();
      uart_TxChar(ch);
} }
```