## 32 Bit Multiplication

```
    AREA MULTIPLICATION32MEMORY, CODE, READONLY
    EXPORT __main

NUM1 DCD 0x12345678
NUM2 DCD 0x11111111

__main
    LDR R0,NUM1
    LDR R1,NUM2
    UMULL R3,R2,R0,R1
    LDR R4,=RESULT1
    LDR R5,=RESULT2
    STR R3,[R4]
    STR R2,[R5]

STOP B STOP

    AREA DATA1, DATA, READWRITE

RESULT1 DCD 0x0
RESULT2 DCD 0x0

    END
```

## 16 Bit Multiplication

```
    AREA MULTIPLICATION, CODE, READONLY
    EXPORT __main

NUM1 DCD 0x0002
NUM2 DCD 0x0003

__main
    LDR R0,NUM1
    LDR R1,NUM2
    MUL R2,R1,R0

STOP B STOP
    END
```

## Sum of first 10  numbers

```
    AREA SUMM10NUMBERS, CODE, READONLY
    EXPORT __main

__main
    MOV R0,#10
    MOV R1,#0
LOOP
    ADD R1,R0
    SUBS R0,#1
    BNE LOOP
    LDR R2,=SUM
    STR R1,[R2]

STOP B STOP
    AREA DATA1, DATA, READWRITE

SUM DCD 0x0
    END
```

## Sum Of first 10 numbers without loop

```
    AREA SUMWITHOUTLOOP10NUMBERS, CODE, READONLY
    EXPORT __main

__main
    MOV R0,#10
    MOV R5,#2
    ADD R1,R0,#1
    MUL R3,R1,R0
    UDIV R7,R3,R5
    LDR R6,=SUM
    STR R7,[R6]

STOP B STOP

    AREA DATA1, DATA, READWRITE

SUM DCD 0x0

    END
```

# Factorial of a Number

```
    AREA ECATORIAL, CODE, READONLY
    EXPORT __main

__main
    MOV R1,#1
    MOV R2,#5

LOOP
    MUL R1,R2,R1
    SUBS R2,#1
    BNE LOOP
    LDR R3,=FACT
    STR R1,[R3]

STOP B STOP

    AREA DATA1, DATA, READWRITE

FACT DCD 0x0

    END
```

## 16 bit Array sum in 32bit

```
    AREA ADD16STORE32, CODE, READONLY
    EXPORT __main

__main
    LDR R0, =COUNT      ; Load the address of COUNT into R0
    LDR R1, =ARRAY16    ; Load the address of ARRAY16 into R1
    MOV R2, #0          ; Initialize R2 with 0
    LDR R3, [R0]        ; Load the value at the address stored in R0 into R3

LOOP
    LDRH R4, [R1], #4   ; Load a half-word (16-bit) value from the address
stored in R1 into R4, and increment R1 by 2
    ADD R2, R2, R4      ; Add the value in R4 to R2
    SUBS R3, R3, #1     ; Decrement R3 by 1 and update the flags

    BNE LOOP            ; Branch to LOOP if the result of the subtraction is
not zero

    STR R2, [R1]        ; Store the value in R2 at the address stored in R1

STOP B STOP              ; Infinite loop to halt the program

    AREA INPUT, DATA, READWRITE

COUNT DCD 0x0
ARRAY16 DCD 0x0
    END
```

# ADD TWO 64 BIT NUMBERS

```
    AREA ADDITION64, CODE, READONLY
    EXPORT __main

VAL1 DCD 0x11111111
VAL2 DCD 0x22222222
VAL3 DCD 0x33333333
VAL4 DCD 0x44444444

__main
    LDR R0,VAL1
    LDR R1,VAL2
    LDR R2,VAL3
    LDR R3,VAL4
    ADDS R4,R0,R2
    ADC R5,R1,R3
    LDR R6,=LOWER
    STR R4,[R6]
    LDR R7,=HIGHER
    STR R5,[R7]

STOP B  STOP

    AREA DATA1, DATA, READWRITE

LOWER DCD 0x0
HIGHER DCD 0x0

    END
```

# LOOK UP TABLE

```
    AREA LOOKUP, CODE, READONLY
    EXPORT __main

TABLE1 DCD 0x00000000
    DCD 0x00000001
    DCD 0x00000004
    DCD 0x00000009
    DCD 0x00000010
    DCD 0x00000019
    DCD 0x00000024
    DCD 0x00000031
    DCD 0x00000040
    DCD 0x00000051
    DCD 0x00000064

__main
    LDR R0,=TABLE1
    LDR R1,=8
    MOV R1,R1,LSL#0X2
;   LSL R1, R1, #2
    ADD R0,R0,R1
    LDR R3,[R0]
    NOP
    NOP
```

# LARGEST NUMBER

```
    AREA LARGEST, CODE, READONLY
    EXPORT __main

ARRAY1 DCD 1
    DCD 8
    DCD 3
    DCD 7
    DCD 9

__main
     LDR R0,=ARRAY1
     LDR R1,=5
     LDR R3,[R0],#4
     SUBS R1,R1,#1
LOOP
    LDR R4,[R0],#4
    CMP R3,R4
    BHI LARGER
    MOV R3,R4
LARGER
    SUBS R1,R1,#1
    BNE LOOP
    LDR R2,=LARGENO
    STR R3,[R2]
    NOP
    NOP

    AREA INPUT,DATA, READWRITE
LARGENO DCD 0x0

    END
```

# SMALLEST NUMBER

```
    AREA SMALLEST, CODE, READONLY
    EXPORT __main

ARRAY1 DCD 5
    DCD 4
    DCD 10
    DCD 1
    DCD 8

__main
    LDR R0,=ARRAY1
    LDR R1,=5
    LDR R3,[R0],#4
    SUBS R1,R1,#1
LOOP
    LDR R4,[R0],#4
    CMP R3,R4
    BLS SMALLER
    MOV R3,R4
SMALLER
    SUBS R1,R1,#1
    BNE LOOP
    LDR R5,=SMALLENO
    STR R3,[R5]
    NOP
    NOP

    AREA DATA1, DATA, READWRITE
SMALLENO DCD 0x0
    END
```

# Ascending Order

```
    AREA ASCENDING, CODE, READONLY
    EXPORT __main

__main
    MOV R8,#4
    LDR R2,=CVALUE
    LDR R3,=DVALUE
LOOP0
    LDR R1,[R2],#4
REGION
    STR R1,[R3],#4
    SUBS R8,R8,#1
    CMP R8,#0
    BNE LOOP0
START1
    MOV R5,#3
    MOV R7,#0
    LDR R1,=DVALUE
LOOP
    LDR R2,[R1],#4
    LDR R3,[R1]
    CMP R2,R3
    BLT LOOP2
    STR R2,[R1],#-4
    STR R3,[R1]
    MOV R7,#1
    ADD R1,#4
LOOP2
    SUBS R5,R5,#1
    CMP R5,#0
    BNE LOOP
    CMP R7,#0
    BNE START1
    NOP
    NOP


CVALUE DCD 0x44444444
       DCD 0x11111111
       DCD 0x33333333
       DCD 0x22222222


    AREA DATA1, DATA, READWRITE

DVALUE DCD 0x00000000
```

## Descending Order

```
    AREA ASCENDING, CODE, READONLY
    EXPORT __main

__main
    MOV R8,#4
    LDR R2,=CVALUE
    LDR R3,=DVALUE
LOOP0
    LDR R1,[R2],#4
REGION
    STR R1,[R3],#4
    SUBS R8,R8,#1
    CMP R8,#0
    BNE LOOP0
START1
    MOV R5,#3
    MOV R7,#0
    LDR R1,=DVALUE
LOOP
    LDR R2,[R1],#4
    LDR R3,[R1]
    CMP R2,R3
    BGT LOOP2
    STR R2,[R1],#-4
    STR R3,[R1]
    MOV R7,#1
    ADD R1,#4
LOOP2
    SUBS R5,R5,#1
    CMP R5,#0
    BNE LOOP
    CMP R7,#0
    BNE START1
    NOP
    NOP


CVALUE DCD 0x44444444
       DCD 0x11111111
       DCD 0x33333333
       DCD 0x22222222


    AREA DATA1, DATA, READWRITE

DVALUE DCD 0x00000000
```

## Zeroes and Ones in two consecutive memory location

```
    AREA ZEROESONES, CODE, READONLY
    EXPORT __main

__main
    LDR R0,=ARRAY
    MOV R1,#1
    MOV R4,#0
    MOV R5,#0

LOOP
    LDR R2,[R0],#4
    MOV R3,#32

LOOP0
    MOVS R2,R2,ROR#1
    BHI SETONE
    B SETZERO

SETONE
    ADD R4,#1
    B NEXT

SETZERO
    ADD R5,#1
    B NEXT

NEXT
    SUBS R3,#1
    CMP R3,#0
    BNE LOOP0

    SUBS R1,#1
    CMP R1,#0
    BNE LOOP

    LDR R6,=ONES
    STR R4,[R6]
    LDR R7,=ZEROES
    STR R5,[R7]

ARRAY DCD 0x00000002

    AREA DATA1, DATA, READWRITE
ONES DCD 0x0
ZEROES DCD 0x0
    END
```

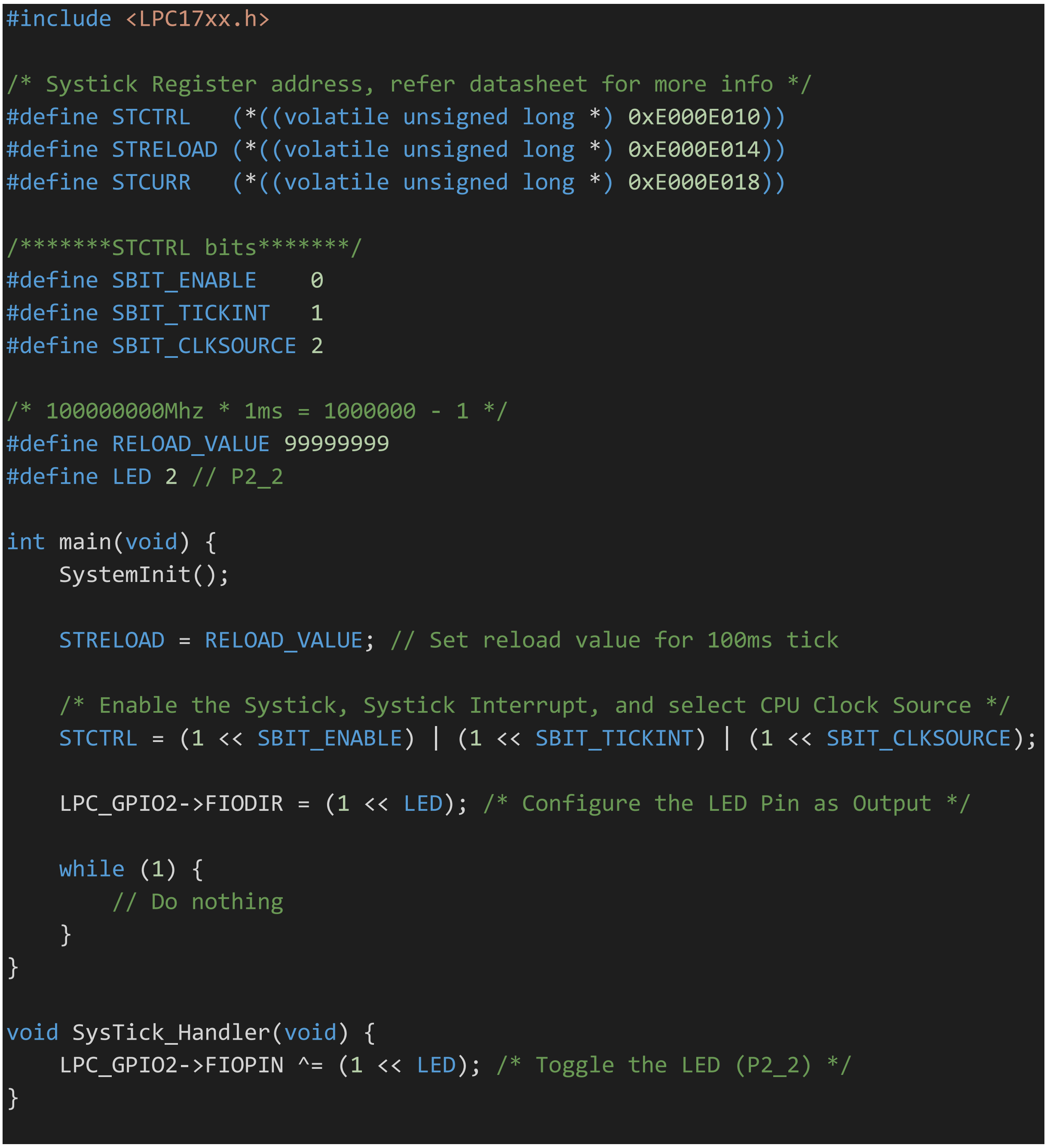
## 32 bit numbers negative

```
    AREA NEGATIVENUMBERS, CODE, READONLY
    EXPORT __main

__main
    MOV R5,#6
    MOV R2,#0
    LDR R4,=VALUE

LOOP
    LDR R1,[R4],#4
    CMP R1,#0
    BLT FOUND
;   ANDS R1,R1,1<<31
;   BHI FOUND
    B LOOP1

FOUND
    ADD R2,R2,#1
    B LOOP1

LOOP1
    SUBS R5,R5,#1
    CMP R5,#0
    BNE LOOP
    NOP
    NOP

VALUE DCD 0x12345678
      DCD 0x8D489867
      DCD 0x33333333
      DCD 0xE605546C
      DCD 0xAAAAAAAA
      DCD 0x99999999
```

## LED with Software Delay

```
#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++)
        {
            // Delay loop
        }
    }
}

int main(void)
{
    SystemInit();                            // Clock and PLL configuration

    LPC_PINCON->PINSEL4 = 0xffffffff;        // Configure the PORT2 Pins as
GPIO
    LPC_GPIO2->FIODIR = 0xffffffff;          // Configure the PORT2 pins as
OUTPUT

    while (1)
    {
        LPC_GPIO2->FIOSET = 0xffffffff;      // Make all the Port pins high
        delay_ms(10);

        LPC_GPIO2->FIOCLR = 0xffffffff;      // Make all the Port pins low
        delay_ms(10);
    }
}
```

## Led Delay using Systick timer



```
#include <LPC17xx.h>

/* Systick Register address, refer datasheet for more info */
#define STCTRL   (*((volatile unsigned long *) 0xE000E010))
#define STRELOAD (*((volatile unsigned long *) 0xE000E014))
#define STCURR   (*((volatile unsigned long *) 0xE000E018))

/*******STCTRL bits*******/
#define SBIT_ENABLE    0
#define SBIT_TICKINT   1
#define SBIT_CLKSOURCE 2

/* 100000000Mhz * 1ms = 1000000 - 1 */
#define RELOAD_VALUE 99999999
#define LED 2 // P2_2

int main(void) {
    SystemInit();

    STRELOAD = RELOAD_VALUE; // Set reload value for 100ms tick

    /* Enable the Systick, Systick Interrupt, and select CPU Clock Source */
    STCTRL = (1 << SBIT_ENABLE) | (1 << SBIT_TICKINT) | (1 << SBIT_CLKSOURCE);

    LPC_GPIO2->FIODIR = (1 << LED); /* Configure the LED Pin as Output */

    while (1) {
        // Do nothing
    }
}

void SysTick_Handler(void) {
    LPC_GPIO2->FIOPIN ^= (1 << LED); /* Toggle the LED (P2_2) */
}
```

## Led using switch by polling

```
#include <lpc17xx.h>

#define SwitchPinNumber 11
#define LedPinNumber 0

/* Start the main program */
int main()
{
    uint32_t switchStatus;

    SystemInit(); /* Clock and PLL configuration */

    LPC_PINCON->PINSEL4 = 0x000000; /* Configure the Pins for GPIO */

    /* Configure the LED pin as output and SwitchPin as input */
    LPC_GPIO2->FIODIR = ((1 << LedPinNumber) | (0 << SwitchPinNumber));

    while (1)
    {
        /* Turn On all the LEDs and wait for one second */
        switchStatus = (LPC_GPIO2->FIOPIN >> SwitchPinNumber) & 0x01; /* Read the switch status */

        if (switchStatus == 1)
        {
            /* Turn ON/OFF LEDs depending on switch status */
            LPC_GPIO2->FIOPIN = (1 << LedPinNumber); /* Turn on the LED */
        }
        else
        {
            LPC_GPIO2->FIOPIN = (0 << LedPinNumber); /* Turn off the LED */
        }
    }
}
```

## PLL ON

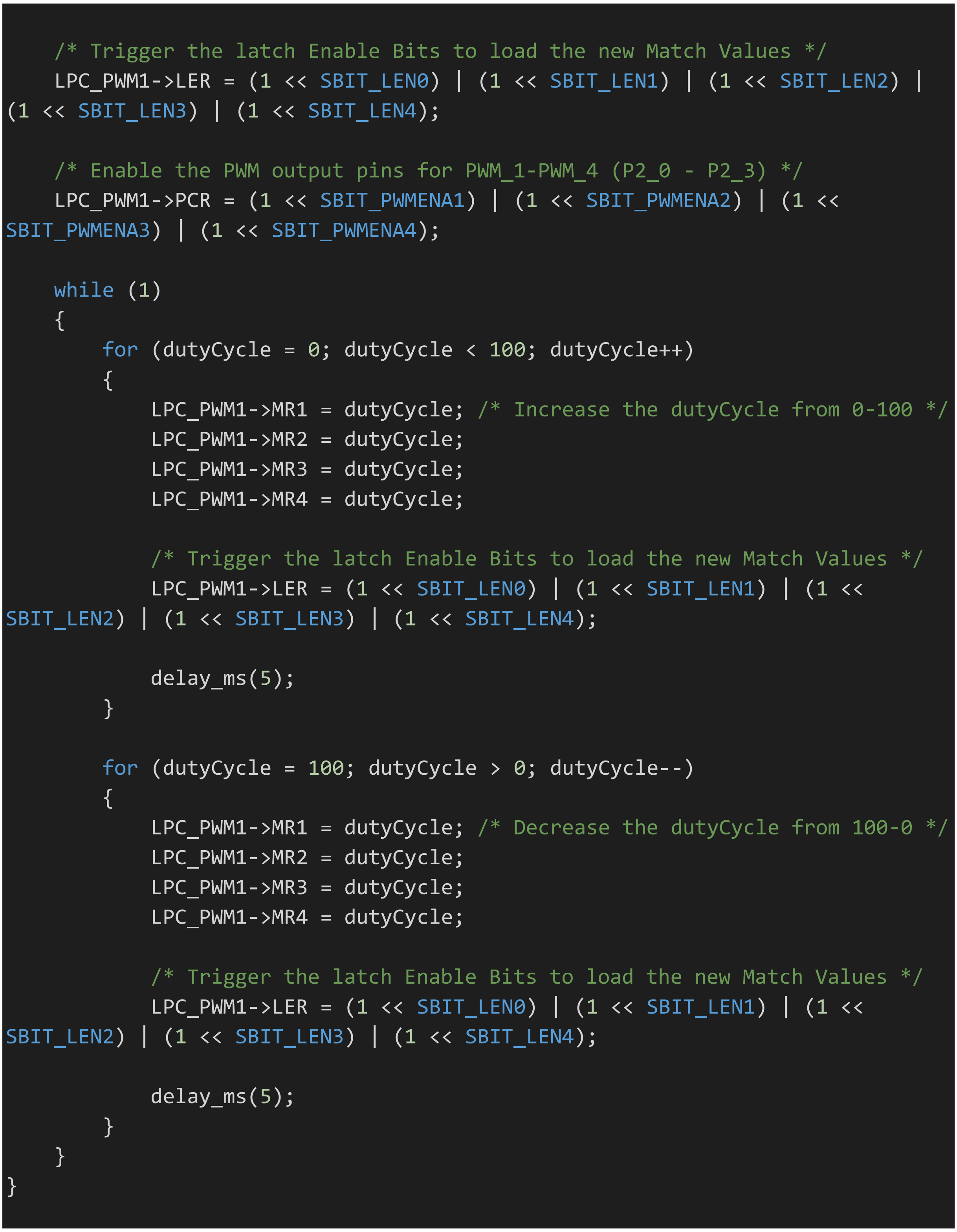
```c
#include <LPC17xx.h>

#define CCLKCFG (*(volatile unsigned long *)(0x400FC104))
#define PLL0CON (*(volatile unsigned long *)(0x400FC080))
#define PLL0FEED (*(volatile unsigned long *)(0x400FC08C))
#define PLL0STAT (*(volatile unsigned long *)(0x400FC088))
#define PLL0CFG (*(volatile unsigned long *)(0x400FC084))

// Function prototypes
void delay(void);

int main() {
    LPC_GPIO2->FIODIR |= 0x0000007C;

    // CCLKCFG=0x000000EE; // divider divides by this number plus 1

    // Set PLL0 multiplier
    PLL0CFG = 0x0015013A; // Arbitrary multiply value, divide value left at 1
    PLL0FEED = 0x000000AA; // Feed the PLL
    PLL0FEED = 0x00000055;

    // Turn on PLL0
    PLL0CON |= 1 << 0;
    PLL0FEED = 0x000000AA; // Feed the PLL
    PLL0FEED = 0x00000055;

    // Wait for main PLL (PLL0) to come up
    while ((PLL0STAT & (1 << 24)) == 0x00);

    // Wait for PLOCK0 to become 1
    while ((PLL0STAT & (1 << 26)) == 0x00);

    // Connect to the PLL0
    PLL0CON |= 1 << 1;
    PLL0FEED = 0x000000AA; // Feed the PLL
    PLL0FEED = 0x00000055;

    while ((PLL0STAT & (1 << 25)) == 0x00); // Wait for PLL0 to connect

    while (1) {
        LPC_GPIO2->FIOPIN ^= (0x0000007C);
        delay();
    }
}

void delay(void) {
    // Delay function.
    int j; // Loop variable j
    for (j = 0; j < 50000; j++) {
        j++;
        j--; // Waste time
    }
}
```

## PLL OFF

```
#include <LPC17xx.h>

#define CCLKCFG (*(volatile unsigned char *)(0x400FC104))
#define PLL0CON (*(volatile unsigned char *)(0x400FC080))
#define PLL0FEED (*(volatile unsigned char *)(0x400FC08C))
#define PLL0STAT (*(volatile unsigned char *)(0x400FC088))

// Function prototypes
void delay(void);

int main() {
    LPC_GPIO2->FIODIR |= 0x0000007C;
    CCLKCFG = 0x000000FF;

    // Disconnect PLL0
    PLL0CON &= !(1 << 1); // Clears bit 1 of PLL0CON, the Connect bit
    PLL0FEED = 0xAA; // Feed the PLL. Enables action of the above line
    PLL0FEED = 0x55;

    // Wait for PLL0 to disconnect. Wait for bit 25 to become 0.
    while ((PLL0STAT & (1 << 25)) != 0x00); // Bit 25 shows connection status

    // Turn off PLL0; on completion, PLL0 is bypassed.
    PLL0CON &= !(1 << 0); // Bit 0 of PLL0CON disables PLL
    PLL0FEED = 0xAA; // Feed the PLL. Enables action of the above line
    PLL0FEED = 0x55;

    // Wait for PLL0 to shut down
    while ((PLL0STAT & (1 << 24)) != 0x00); // Bit 24 shows enable status

    /**** Insert Optional Extra Code Here ****
    to change PLL0 settings or clock source.
    ** OR ** just continue with PLL0 disabled and bypassed */

    // Blink at the new clock frequency
    while (1) {
        LPC_GPIO2->FIOPIN ^= (0x0000007C);
        delay();
    }
}

void delay(void) {
    // Delay function.
    int j; // Loop variable j
    for (j = 0; j < 5000000; j++) {
        j++;
        j--; // Waste time
    }
}
```

## PWM Model of Arm Controller

```
#include <lpc17xx.h>

void delay_ms(unsigned int ms)
{
    unsigned int i, j;
    for (i = 0; i < ms; i++)
    {
        for (j = 0; j < 50000; j++);
    }
}

#define SBIT_CNTEN 0
#define SBIT_PWMEN 2
#define SBIT_PWMMR0R 1
#define SBIT_LEN0 0
#define SBIT_LEN1 1
#define SBIT_LEN2 2
#define SBIT_LEN3 3
#define SBIT_LEN4 4
#define SBIT_PWMENA1 9
#define SBIT_PWMENA2 10
#define SBIT_PWMENA3 11
#define SBIT_PWMENA4 12
#define PWM_1 0 // P2_0 (0-1 Bits of PINSEL4)
#define PWM_2 2 // P2_1 (2-3 Bits of PINSEL4)
#define PWM_3 4 // P2_2 (4-5 Bits of PINSEL4)
#define PWM_4 6 // P2_3 (6-7 Bits of PINSEL4)

int main(void)
{
    int dutyCycle;
    SystemInit();

    /* Configure pins (P2_0 - P2_3) for PWM mode. */
    LPC_PINCON->PINSEL4 = (1 << PWM_1) | (1 << PWM_2) | (1 << PWM_3) | (1 <<
PWM_4);

    /* Enable Counters, PWM module */
    LPC_PWM1->TCR = (1 << SBIT_CNTEN) | (1 << SBIT_PWMEN);
    LPC_PWM1->PR = 0x0; /* No Prescalar */
    LPC_PWM1->MCR = (1 << SBIT_PWMMR0R); /* Reset on PWMMR0, reset TC if it
matches MR0 */
    LPC_PWM1->MR0 = 100; /* Set PWM cycle (Ton+Toff)=100) */
    LPC_PWM1->MR1 = 50; /* Set 50% Duty Cycle for all four channels */
    LPC_PWM1->MR2 = 50;
    LPC_PWM1->MR3 = 50;
    LPC_PWM1->MR4 = 50;
```



```
    /* Trigger the latch Enable Bits to load the new Match Values */
    LPC_PWM1->LER = (1 << SBIT_LEN0) | (1 << SBIT_LEN1) | (1 << SBIT_LEN2) |
(1 << SBIT_LEN3) | (1 << SBIT_LEN4);

    /* Enable the PWM output pins for PWM_1-PWM_4 (P2_0 - P2_3) */
    LPC_PWM1->PCR = (1 << SBIT_PWMENA1) | (1 << SBIT_PWMENA2) | (1 <<
SBIT_PWMENA3) | (1 << SBIT_PWMENA4);

    while (1)
    {
        for (dutyCycle = 0; dutyCycle < 100; dutyCycle++)
        {
            LPC_PWM1->MR1 = dutyCycle; /* Increase the dutyCycle from 0-100 */
            LPC_PWM1->MR2 = dutyCycle;
            LPC_PWM1->MR3 = dutyCycle;
            LPC_PWM1->MR4 = dutyCycle;

            /* Trigger the latch Enable Bits to load the new Match Values */
            LPC_PWM1->LER = (1 << SBIT_LEN0) | (1 << SBIT_LEN1) | (1 <<
SBIT_LEN2) | (1 << SBIT_LEN3) | (1 << SBIT_LEN4);

            delay_ms(5);
        }

        for (dutyCycle = 100; dutyCycle > 0; dutyCycle--)
        {
            LPC_PWM1->MR1 = dutyCycle; /* Decrease the dutyCycle from 100-0 */
            LPC_PWM1->MR2 = dutyCycle;
            LPC_PWM1->MR3 = dutyCycle;
            LPC_PWM1->MR4 = dutyCycle;

            /* Trigger the latch Enable Bits to load the new Match Values */
            LPC_PWM1->LER = (1 << SBIT_LEN0) | (1 << SBIT_LEN1) | (1 <<
SBIT_LEN2) | (1 << SBIT_LEN3) | (1 << SBIT_LEN4);

            delay_ms(5);
        }
    }
}
```

## LED using switch by interrupt method

```
#include <lpc17xx.h>

#define PINSEL_EINT0 20
#define PINSEL_EINT1 22
#define LED1 0
#define LED2 1
#define SBIT_EINT0 0
#define SBIT_EINT1 1
#define SBIT_EXTMODE0 0
#define SBIT_EXTMODE1 1
#define SBIT_EXTPOLAR0 0
#define SBIT_EXTPOLAR1 1

void EINT0_IRQHandler(void)
{
    LPC_SC->EXTINT = (1 << SBIT_EINT0); /* Clear Interrupt Flag */
    LPC_GPIO2->FIOPIN ^= (1 << LED1);  /* Toggle LED1 every time INTR0 is
generated */
}

void EINT1_IRQHandler(void)
{
    LPC_SC->EXTINT = (1 << SBIT_EINT1); /* Clear Interrupt Flag */
    LPC_GPIO2->FIOPIN ^= (1 << LED2);  /* Toggle LED2 every time INTR1 is
generated */
}

int main()
{
    SystemInit();

    LPC_SC->EXTINT = (1 << SBIT_EINT0) | (1 << SBIT_EINT1); /* Clear Pending
interrupts */

    LPC_PINCON->PINSEL4 = (1 << PINSEL_EINT0) | (1 << PINSEL_EINT1); /*
Configure P2_10, P2_11 as EINT0/1 */

    LPC_SC->EXTMODE = (1 << SBIT_EXTMODE0) | (1 << SBIT_EXTMODE1);   /*
Configure EINTx as Edge Triggered */
    LPC_SC->EXTPOLAR = (1 << SBIT_EXTPOLAR0) | (1 << SBIT_EXTPOLAR1); /*
Configure EINTx as Falling Edge */

    LPC_GPIO2->FIODIR = (1 << LED1) | (1 << LED2); /* Configure LED pins as
OUTPUT */
    LPC_GPIO2->FIOPIN = 0x00;

    NVIC_EnableIRQ(EINT0_IRQn); /* Enable the EINT0, EINT1 interrupts */
    NVIC_EnableIRQ(EINT1_IRQn);

    while (1)
    {
        // Do nothing
    }
}
```

# UART

```
#include <LPC17xx.h>

void delay(unsigned int r1);
void UART0_Init(void);
void UART0_IRQHandler(void);

unsigned long int r = 0, i = 0;
unsigned char tx0_flag = 0;
unsigned char *ptr, arr[] = "Hello world";

int main(void)
{
    SystemInit();
    SystemCoreClockUpdate();
    UART0_Init();

    while (1)
    {
        ptr = arr;

        while (*ptr != '\0')
        {
            LPC_UART0->THR = *ptr++;
            while (tx0_flag == 0x00);
            tx0_flag = 0x00;
            for (i = 0; i < 200; i++);
        }

        for (i = 0; i < 500; i++)
            delay(625); // Delay
    }
}

void UART0_Init(void)
{
    LPC_SC->PCONP |= 0x00000008;     // UART0 peripheral enable
    LPC_PINCON->PINSEL0 |= 0x00000050;    // Selecting TX0[P0.2-->5:4] and RX0[P0.3-->7:6] of UART0
    LPC_UART0->LCR = 0x00000083;     // Enable divisor latch, parity disable, 1 stop bit, 8-bit word length line control register
    LPC_UART0->DLM = 0X00;
    LPC_UART0->DLL = 0x13;  // Select baud rate 9600 bps
    LPC_UART0->LCR = 0X00000003;
    LPC_UART0->FCR = 0x07;
    LPC_UART0->IER = 0X03;  // Select Transmit and receive interrupt
    NVIC_EnableIRQ(UART0_IRQn);  // Assigning channel
}
```

```
void UART0_IRQHandler(void)
{
    unsigned long Int_Stat;
    Int_Stat = LPC_UART0->IIR;  // Reading the data from interrupt
identification register
    Int_Stat = Int_Stat & 0x06;  // Masking other than transmit int & receive
data indicator

    if ((Int_Stat & 0x02) == 0x02)  // Transmit interrupt
        tx0_flag = 0xff;
}

void delay(unsigned int r1)
{
    for (r = 0; r < r1; r++);
}
```

**VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY**

| B.Tech. VI Semester | L | T/P/D | C |
|---|---|---|---|
| | 0 | 2 | 1 |

## (19PC2CS78) EMBEDDED SYSTEMS DESIGN LABORATORY

**COURSE OBJECTIVES:**

- To introduce the principles involved in the design and implementation of embedded systems
- To provide familiarity with the basic concepts and terminology of the target area, the embedded systems design flow
- To introduce the embedded system architecture
- To introduce the methods of executive device control and testing

**COURSE OUTCOMES:** After completion of the course, the student should be able to

**CO-1:** Develop assembly language and high-level language programming skills to microprocessors and microcontrollers-based systems

**CO-2:** install, configure and utilize tool sets for developing applications based on ARM processor core

**CO-3:** develop prototype codes using commonly available on and off chip peripherals on the Cortex M3

**CO-4:** Propose, design and implement the ideas for measuring, controlling various physical parameters of real-world problems

**PART A:**

Conduct the following experiments by writing Assembly Language Program (ALP) using ARM Cortex M3 Registers using an evaluation board/simulator and the required software tool.

1. Write an ALP to multiply two 16-bit binary numbers.
2. Write an ALP to find the sum of first 10 integer numbers.
3. Write an ALP to find factorial of a number.
4. Write an ALP to add an array of 16-bit numbers and store the 32-bit result in internal RAM
5. Write an ALP to add two 64-bit numbers.
6. Write an ALP to find the square of a number (1 to 10) using look-up table.
7. Write an ALP to find the largest/smallest number in an array of 32 numbers.
8. Write an ALP to arrange a series of 32-bit numbers in ascending/descending order.
9. Write an ALP to count the number of ones and zeros in two consecutive memory locations.
10. Write an ALP to Scan a series of 32-bit numbers to find how many are negative.

**PART B:**

Conduct the following experiments on an ARM CORTEX M3 evaluation board using evaluation version of Embedded 'C' & Keil Uvision-4 tool/compiler.

1. Blink an LED with software delay.
2. Blink an LED with delay generated using the SysTick timer.
3. System clock real time alteration using the PLL modules.
4. Using the Internal PWM module of ARM controller generate PWM and vary its duty cycle.
5. Control an LED using switch by polling method

6. Control an LED using switch by interrupt method
7. Display "Hello World" message using Internal UART