**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**

#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++);

}

