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**

#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(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 as high \*/

delay\_ms(10);

LPC\_GPIO2->FIOCLR = 0xffffffff; /\* Make all the Port pins as low \*/

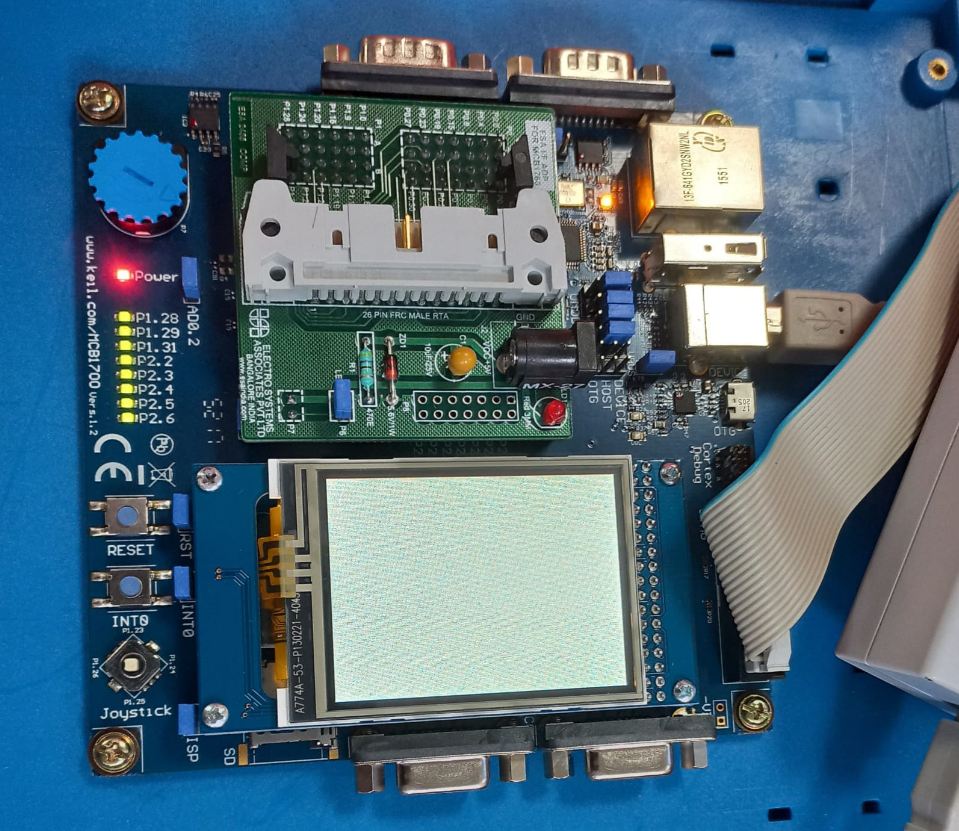
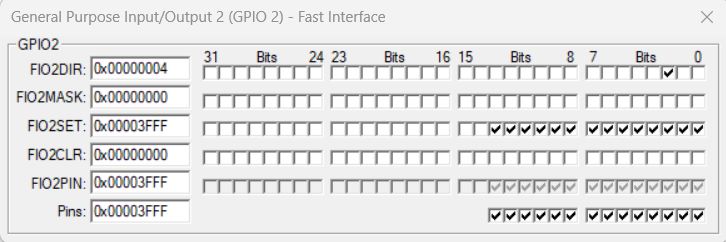
delay\_ms(10);

}

}

Output:

In Peripherals 🡺 GPIO pins 🡺 check pin 2 to see the pins.

**2. Blink an LED with delay generated using the 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; // Reload value for 100ms tick

/\* Enable the Systick, Systick Interrup 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

}

}

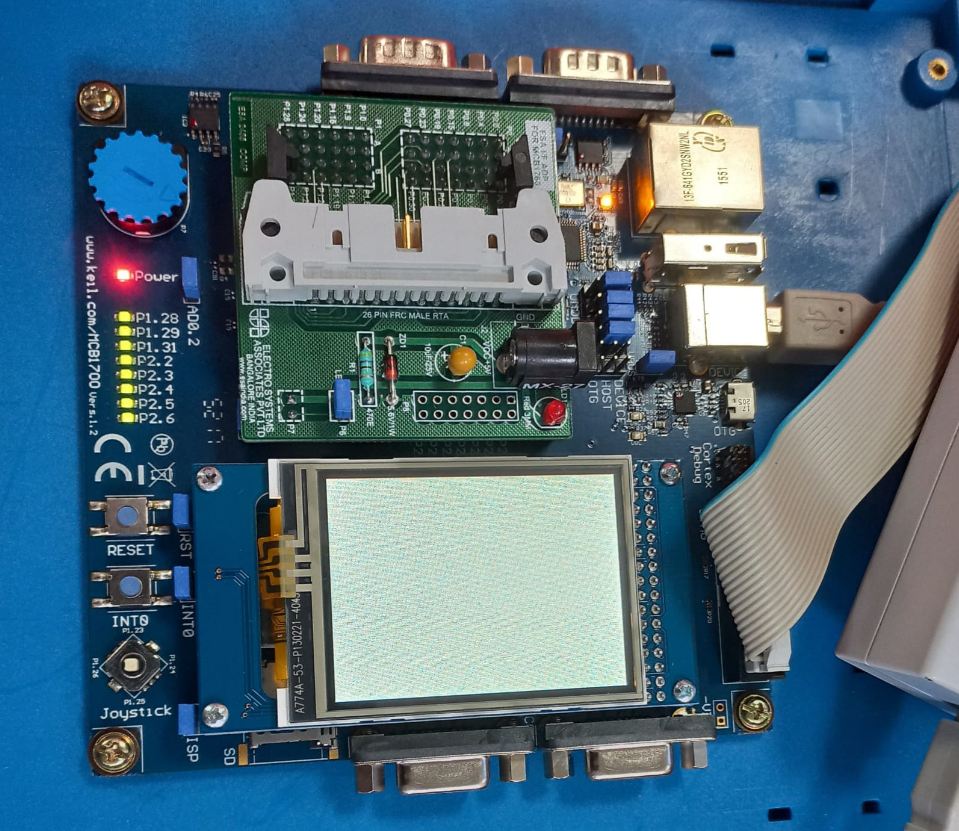
void SysTick\_Handler(void)

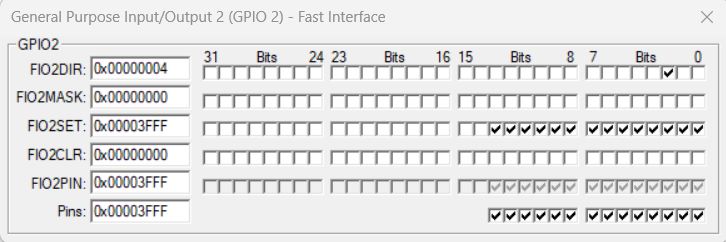
{

LPC\_GPIO2->FIOPIN ^= (1<<LED); /\* Toggle the LED1 (P2\_0) \*/

}

Output:





**3. System clock real time alteration using the PLL modules.**

#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

}

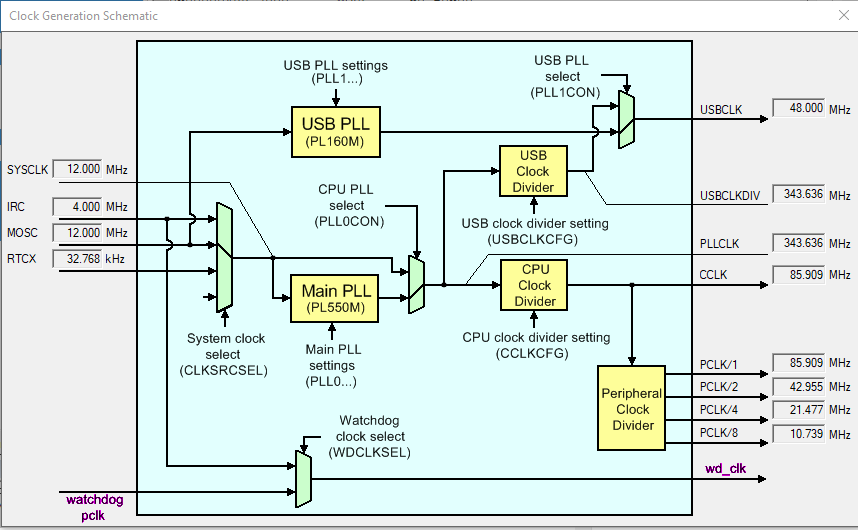
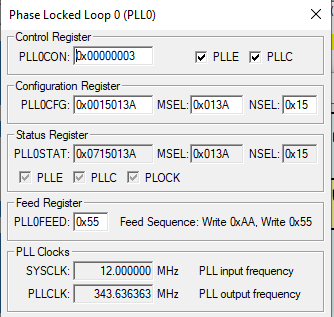
}

Output:

In options panel go to debug panel, select option ‘use JLINK’ and change text to DARMP1.sll and on right side TARMP1.sll

Go to peripherals 🡺 System clock 🡺 PLL 0

When PLL is On , the SYSCLK and PLLCLK are different and PLLCLK>SYSCLK.



**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 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 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

}

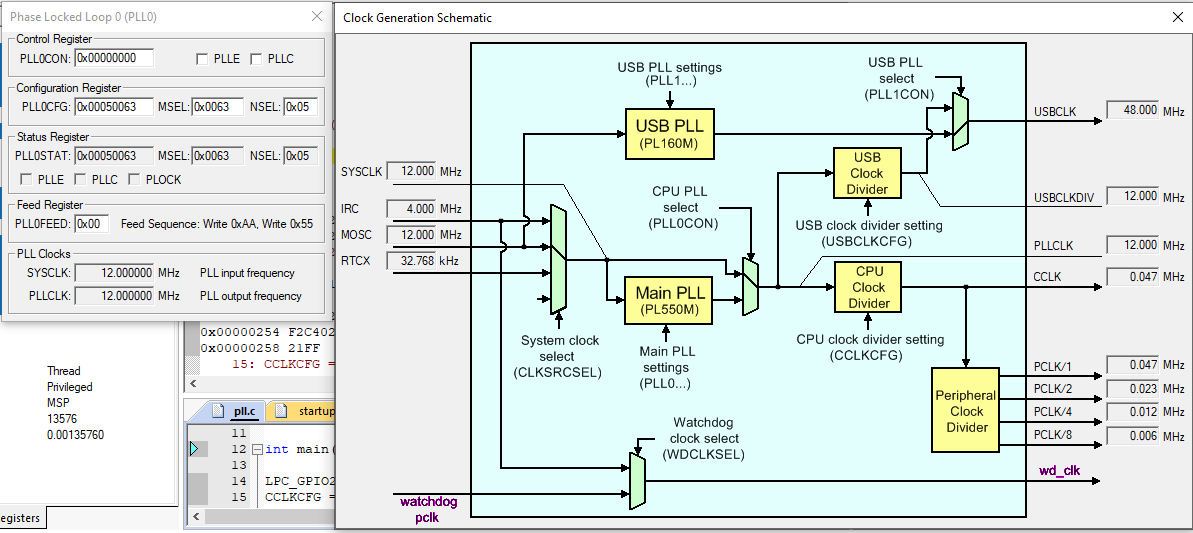
}

Output:

In options panel go to debug panel, select option ‘use JLINK’ and change text to DARMP1.sll and on right side TARMP1.sll

Go to peripherals 🡺 System clock 🡺 PLL 0

When PLL is Off , the SYSCLK and PLLCLK are same and PLLCLK=SYSCLK.



**4. Using the Internal PWM module of ARM controller generate PWM and vary its duty cycle.**

#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();

/\* Cofigure 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);

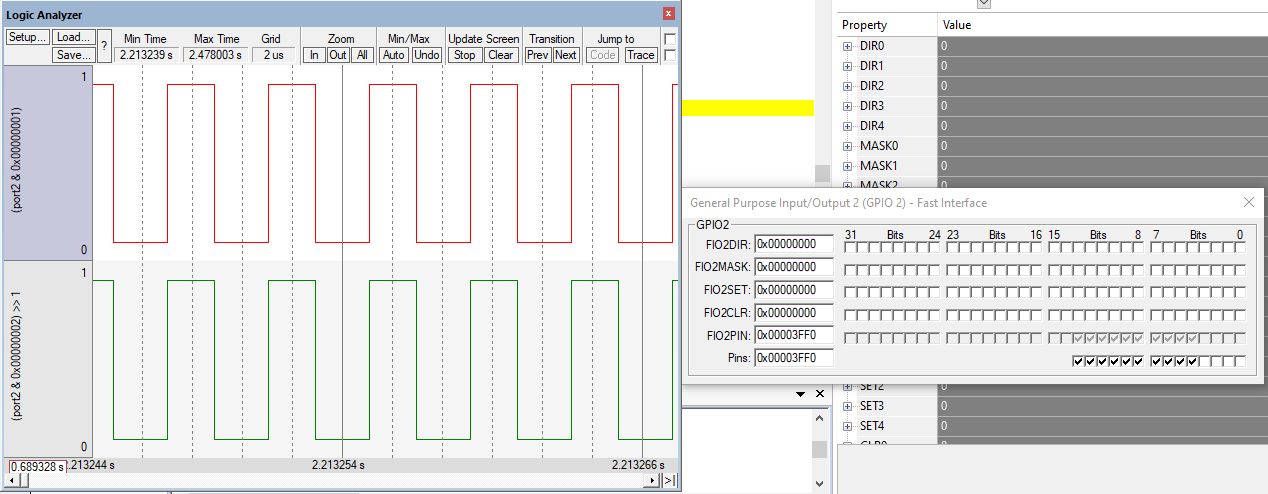
}

}

}

Output:

Go to logic Analyzer (in View Panel) 🡺 setup🡺add pins by select add pin icon🡺 ‘port 2.1,2.2,2.3…’ and start analyser to see the below output. And disable and enable pin 11 to blink the LED. Change the display type to ‘bit’ in dropdown.



**5. Control an LED using switch by polling method**

#include <lpc17xx.h>

#define SwitchPinNumber 11

#define LedPinNumber 2

/\* 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);

}

else

{

LPC\_GPIO2->FIOPIN = (0<<LedPinNumber);

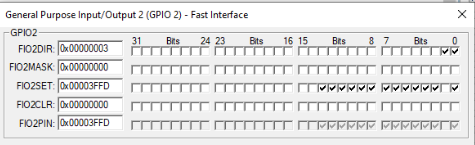
}

}

}

Output:

Mask pin 11(tick in FIO2MASK Column) to switch on/off the LED . Go to peripherals🡺GPIO Pins🡺Pin2



**6. Control an 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 the LED1 everytime INTR0 is generated \*/

}

void EINT1\_IRQHandler(void)

{

LPC\_SC->EXTINT = (1<<SBIT\_EINT1); /\* Clear Interrupt Flag \*/

LPC\_GPIO2->FIOPIN ^= (1<< LED2); /\* Toggle the LED2 everytime 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\_EXTPOLAR0); /\* 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)

{

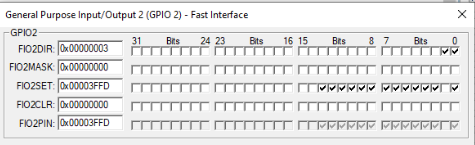
// Do nothing

}

}

Output:

Go to peripherals🡺GPIO Pins🡺Pin2



**7. Display “Hello World” message using Internal 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; //for 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, 8bit 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 txmit int & rcve data indicator

if((Int\_Stat & 0x02)== 0x02) //transmit interrupt

tx0\_flag = 0xff;

}

void delay(unsigned int r1)

{

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

}

Output:

Go to View 🡺 Serial Window 🡺UART #1

