Week 3 Lab Work

Q1. Write a C Code to configure Watchdog Timer.

# include <LPC17xx.h>

# include "lcd.h"

#define ALL\_LED (0xFF << 19)

#define BUZZER (1 << 27)

int main(){

LPC\_GPIO1 -> FIODIR |= ALL\_LED;

LPC\_GPIO1 -> FIODIR |= BUZZER;

LPC\_GPIO1 -> FIOCLR = ALL\_LED;

LPC\_GPIO1 -> FIOCLR = BUZZER;

LPC\_WDT -> WDMOD |= (1 << 1) | (1 << 0); //ENABLE AND RESET ON on TIME-OUT

LPC\_WDT -> WDCLKSEL &= ~0x03; //SELECT INTERNAL RC OSCILLATOR FOR CLK

LPC\_WDT -> WDTC = 4000000; //DELAY = COUNT\*CLK = 4000000 \* (0.25u\*4) = 4sec

LPC\_WDT -> WDFEED = 0xAA; //WRITE FEED SEQUENCE TO START WDT

LPC\_WDT -> WDFEED = 0x55;

LPC\_GPIO1 -> FIOSET = BUZZER;

delay(100);

LPC\_GPIO1 -> FIOCLR = BUZZER;

while(1){

LPC\_GPIO1 -> FIOSET = ALL\_LED;

delay(500);

LPC\_GPIO1 -> FIOCLR = ALL\_LED;

delay(500);

while(1);

//RELOADING WDT IN 4sec

//LPC\_WDT -> WDFEED = 0xAA; //reload WDT before time-out

//LPC\_WDT -> WDFEED = 0x55;

}

}

Q2. Write a C Code to configure Pulse-width by PWM (Pulse-Width Modulation) for Single Edge Mode.

# include <LPC17xx.h>

# include "lcd.h"

int main(){

LPC\_SC -> PCONP |= (1 << 6); //POWER ON PWM1

LPC\_PINCON -> PINSEL3 |= (1 << 9);

LPC\_PINCON -> PINSEL3 &= ~(1 << 8);

LPC\_PWM1 -> PCR |= (1 << 10); //EMABLE PWM1.2 TO GIVE OUTPUT

LPC\_PWM1 -> PCR &= ~(1 << 2); //SINGLE EDGE MODE FOR PWM1.2

LPC\_PWM1 -> PR = 0;

LPC\_PWM1 -> MCR |= (1 << 1); //REPEAT COUNT ON MR0 MATCH

LPC\_PWM1 -> MR0 = 10000; //PWM SINGAL FREQ 100Hz

LPC\_PWM1 -> MR2 = 2000; //20% DUTY CYCLE

LPC\_PWM1 -> LER |= (1 << 2) | (1 << 0); //LOAD ENABLE FOR MR2 & MR0

LPC\_PWM1 -> TCR |= (1 << 3) | (1 << 0); //0 FOR ENABLE COUNTER & 3 FOR ENABLE PWM

while(1){

//20% duty cycle

LPC\_PWM1 -> MR2 = 2000;

LPC\_PWM1 -> LER = (1 << 2);

delay(400);

//40% duty cycle

LPC\_PWM1 -> MR2 = 2000;

LPC\_PWM1 -> LER = (1 << 2);

delay(400);

//60% duty cycle

LPC\_PWM1 -> MR2 = 6000;

LPC\_PWM1 -> LER = (1 << 2);

delay(400);

//80% duty cycle

LPC\_PWM1 -> MR2 = 8000;

LPC\_PWM1 -> LER = (1 << 2);

delay(400);

}

}

Q3. Write a C Code to configure Pulse-width by PWM (Pulse-Width Modulation) for Double Edge Mode.

#include<LPC17xx.h>

#include<stdint.h>

# include "lcd.h"

int main(){

LPC\_SC -> PCONP |= (1<<6); //POWER ON PWM1.

LPC\_PINCON -> PINSEL3 |= (1<<9); //SELECT ALTERNATE POWER-ON PWM1 P1.20

LPC\_PINCON -> PINSEL3 &= ~(1<<8);

LPC\_PWM1 -> PCR |= (1<<10); //ENABLE PWM1.2 channel output

LPC\_PWM1 -> PCR |= (1<<2); //SELECT DOUBLE EDGE mode for pwm1.2 channnel 2

LPC\_PWM1 -> PR = 0;

LPC\_PWM1 -> MCR |= (1<<1); //REPEAT COUNT ON MR0 MATCH

LPC\_PWM1 -> MR0 = 10000; //PWM SIGNAL FREQUENCY 100 HZ = 1/TIME PERIOD

LPC\_PWM1 -> MR1 = 2000; //SIGNAL SET-TIME

LPC\_PWM1 -> MR2 = 4000; // SIGNAL RESET-TIME 20% DUTY CYCLE

LPC\_PWM1 -> LER |= (1<<2) | (1<<1) | (1<<0); //LOAD ENABLE

LPC\_PWM1 -> TCR |= (1<<3) | (1<<0); //LOAD PC = 0 AND TC = 0 AND ENABLE COUNTER , SELECT PWM TIMER REGISTER ENABLE

while(1){

LPC\_PWM1 -> MR2 = 2000; //duty cycle - 20%

LPC\_PWM1 -> MR1 = 4000;

LPC\_PWM1 -> LER |= (1<<1) | (1<<2);

delay(500);

LPC\_PWM1 -> MR1 = 2000;

LPC\_PWM1 -> MR2 = 6000; //duty cycle - 40%

LPC\_PWM1 -> LER |= (1<<1) | (1<<2);

delay(500);

LPC\_PWM1 -> MR1 = 2000; //RISING EDGE

LPC\_PWM1 -> MR2 = 8000; //FALLING EDGE duty cycle - 60%

LPC\_PWM1 -> LER |= (1<<1) | (1<<2);

delay(500);

LPC\_PWM1 -> MR1 = 2000;

LPC\_PWM1 -> MR2 = 9000; //duty cycle - 70%

LPC\_PWM1 -> LER |= (1<<1) | (1<<2);

delay(500);

}

}

Q4. Write a C Code to configure UART to transmit a single character.

# include <LPC17xx.h>

# include "lcd.h"

int main(){

char ch = 'A';

LPC\_SC -> PCONP |= (1 << 3); //POWER ON UART0 MODULE

LPC\_PINCON -> PINSEL0 |= (1 << 4); //SELECT P0.2 PIN FOR TXD

LPC\_PINCON -> PINSEL0 &= ~(1 << 5);

LPC\_PINCON -> PINSEL0 |= (1 << 6); //SELECT P0.3 PIN FOR TXD

LPC\_PINCON -> PINSEL0 &= ~(1 << 7);

LPC\_UART0 -> LCR |= (1 << 1) | (1 << 0); //DATA LENGTH = 8BITS

LPC\_UART0 -> LCR &= ~(1 << 2); //SINGLE STOP BIT

LPC\_UART0 -> LCR &= ~(1 << 3); //NO PARITY BIT

LPC\_UART0 -> LCR |= (1 << 7); //DIVISOR LATCH ENABLE

LPC\_UART0 -> DLM = 0;

LPC\_UART0 -> DLL = 6; //BAUD-RATE = PCLK/16\*DL = 1000000/16\* (256\*DLM + DLL) = 1000000/(16\*6) = 10400

LPC\_UART0 -> FDR |= (0x0C << 4) | (1 << 0); //BR = PCLK/16\*DL(1+D/M) = 1000000/16\* (256\*DLM+DLL) (1+D/M) = 1000000/(16\*6\*(1+1/12)) = 9615

LPC\_UART0 -> LCR &= ~(1 << 7); //DISABLED LATCH

LPC\_UART0 -> THR = 'A'; //TRANSMITTING VALUE 'A'

while((LPC\_UART0 -> LSR & (1 << 5)) == 0){} //WAIT FOR DATA TRANSMISSION TO COMPLETE

}

Q5. Write a C Code to configure UART to transmit alphabets.

# include <LPC17xx.h>

# include "lcd.h"

int main(){

char ch = 'A';

LPC\_SC -> PCONP |= (1 << 3); //POWER ON UART0 MODULE

LPC\_PINCON -> PINSEL0 |= (1 << 4); //SELECT P0.2 PIN FOR TXD

LPC\_PINCON -> PINSEL0 &= ~(1 << 5);

LPC\_PINCON -> PINSEL0 |= (1 << 6); //SELECT P0.3 PIN FOR TXD

LPC\_PINCON -> PINSEL0 &= ~(1 << 7);

LPC\_UART0 -> LCR |= (1 << 1) | (1 << 0); //DATA LENGTH = 8BITS

LPC\_UART0 -> LCR &= ~(1 << 2); //SINGLE STOP BIT

LPC\_UART0 -> LCR &= ~(1 << 3); //NO PARITY BIT

LPC\_UART0 -> LCR |= (1 << 7); //DIVISOR LATCH ENABLE

LPC\_UART0 -> DLM = 0;

LPC\_UART0 -> DLL = 6; //BAUD-RATE = PCLK/16\*DL = 1000000/16\* (256\*DLM + DLL) = 1000000/(16\*6) = 10400

LPC\_UART0 -> FDR |= (0x0C << 4) | (1 << 0); //BR = PCLK/16\*DL(1+D/M) = 1000000/16\* (256\*DLM+DLL) (1+D/M) = 1000000/(16\*6\*(1+1/12)) = 9615

LPC\_UART0 -> LCR &= ~(1 << 7); //DISABLED LATCH

while(ch <= 'Z') {

LPC\_UART0 -> THR = ch; // Send character

while((LPC\_UART0 -> LSR & (1 << 5)) == 0){} // Wait for data transmission to complete

delay(1000); // Add delay

ch++; // Increment the character

}

while(1);

}

Q6. Write a C Code to configure UART to receive a character.

# include <LPC17xx.h>

# include "lcd.h"

int main(){

char ch;

LPC\_SC -> PCONP |= (1 << 3); //POWER ON UART0 MODULE

lcd\_init();

LPC\_PINCON -> PINSEL0 |= (1 << 4); //SELECT P0.2 PIN FOR TXD

LPC\_PINCON -> PINSEL0 &= ~(1 << 5);

LPC\_PINCON -> PINSEL0 |= (1 << 6); //SELECT P0.3 PIN FOR TXD

LPC\_PINCON -> PINSEL0 &= ~(1 << 7);

LPC\_UART0 -> LCR |= (1 << 1) | (1 << 0); //DATA LENGTH = 8BITS

LPC\_UART0 -> LCR &= ~(1 << 2); //SINGLE STOP BIT

LPC\_UART0 -> LCR &= ~(1 << 3); //NO PARITY BIT

LPC\_UART0 -> LCR |= (1 << 7); //DIVISOR LATCH ENABLE

LPC\_UART0 -> DLM = 0;

LPC\_UART0 -> DLL = 6; //BAUD-RATE = PCLK/16\*DL = 1000000/16\* (256\*DLM + DLL) = 1000000/(16\*6) = 10400

LPC\_UART0 -> FDR |= (0x0C << 4) | (1 << 0); //BR = PCLK/16\*DL(1+D/M) = 1000000/16\* (256\*DLM+DLL) (1+D/M) = 1000000/(16\*6\*(1+1/12)) = 9615

LPC\_UART0 -> LCR &= ~(1 << 7); //DISABLED LATCH

while(1){

while((LPC\_UART0 -> LSR & (1 << 0)) == 0) {} // Wait for data transmission to complete

ch = LPC\_UART0 -> RBR;

data\_reg(ch);

}

}

Q7. Write a C Code to configure UART having header and function file separate to transmit data.

**Uart0.h Code:**

#ifndef UART\_H

#define UART\_H

# include <LPC17xx.h>

# include <stdio.h>

void uart0\_init(void);

void uart0\_byte\_transmit(char byte);

char uart0\_byte\_receive(void);

void uart0\_str\_transmit(char \*str);

void delay(uint32\_t ms);

void new\_line(void);

#endif

**Uart0\_def.c Code:**

# include "uart0.h"

void uart0\_init(void) {

LPC\_SC -> PCONP |= (1 << 3); //POWER ON UART0 MODULE

LPC\_PINCON -> PINSEL0 |= (1 << 4); //SELECT P0.2 PIN FOR TXD

LPC\_PINCON -> PINSEL0 &= ~(1 << 5);

LPC\_PINCON -> PINSEL0 |= (1 << 6); //SELECT P0.3 PIN FOR TXD

LPC\_PINCON -> PINSEL0 &= ~(1 << 7);

LPC\_UART0 -> LCR |= (1 << 1) | (1 << 0); //DATA LENGTH = 8BITS

LPC\_UART0 -> LCR &= ~(1 << 2); //SINGLE STOP BIT

LPC\_UART0 -> LCR &= ~(1 << 3); //NO PARITY BIT

LPC\_UART0 -> LCR |= (1 << 7); //DIVISOR LATCH ENABLE

LPC\_UART0 -> DLM = 0;

LPC\_UART0 -> DLL = 6; //BAUD-RATE = PCLK/16\*DL = 1000000/16\* (256\*DLM + DLL) = 1000000/(16\*6) = 10400

LPC\_UART0 -> FDR |= (0x0C << 4) | (1 << 0); //BR = PCLK/16\*DL(1+D/M) = 1000000/16\* (256\*DLM+DLL) (1+D/M) = 1000000/(16\*6\*(1+1/12)) = 9615

LPC\_UART0 -> LCR &= ~(1 << 7); //DISABLED LATCH

}

void uart0\_byte\_transmit(char byte) {

LPC\_UART0 -> THR = byte; // Send character

while((LPC\_UART0 -> LSR & (1 << 5)) == 0){} // Wait for data transmission to complete

}

char uart0\_byte\_receive(void) {

while((LPC\_UART0 -> LSR & (1 << 0)) == 0) {} // Wait for data transmission to complete

return LPC\_UART0 -> RBR;

}

void uart0\_str\_transmit(char \*str){

while(\*str != '\0'){

uart0\_byte\_transmit(\*str);

str++;

}

}

void delay(uint32\_t ms){

uint32\_t i, j;

for(i = 0; i < ms; i++){

for(j = 0; j < 1250; j++){}

}

}

void new\_line(void){

uart0\_byte\_transmit('\r');

uart0\_byte\_transmit('\n');

}

**Uart\_app.c Code:**

# include "uart0.h"

int main(){

char sval[20];

char ch = 'A';

char str[] = "Sunny";

float cgpa = 8.89;

uart0\_init();

uart0\_byte\_transmit(ch);

new\_line();

uart0\_str\_transmit(str);

new\_line();

sprintf(sval, "%.1f", cgpa);

uart0\_str\_transmit(str);

new\_line();

}

Q8.