ECE IoT 505L – Embedded Programming – Lab Exercises

|  |  |  |
| --- | --- | --- |
| **S.No** | **Experiments** | **Page No** |
| 1. | 64-bit Addition – Assembly | 2 |
| 2. | Memory Block transfer - Assembly | 5 |
| 3. | LED Blinking – C Program | 7 |
| 4. | External Interrupt – C Program | 9 |
| 5. | UART - C Program | 11 |
| 6. | Analog to Digital Converter – C Program | 15 |
| 7. | Real Time Clock – C Program | 19 |
| 8. | FreeRTOS - Task Management | 25 |
| 9. | FreeRTOS- InterTask Communication | 28 |
|  |  |  |
|  |  |  |

**Exercise 1**: Write an assembly program to add two 64 bit numbers in memory.

**Objective**: To write an assembly language program to add the two 64 bit numbers stored in the data memory of LPC1768 (ARM Cortex M3 Microcontroller from NXP).

**Let us consider two 64 bit numbers**

**X = 0x1234567812345678**

**Y = 0x2345678023456789 +**

**--------------------------------------**

**Z =** 0x3579BE013579BE01

**---------------------------------------**

**Update the memory 0x10000100 with the Least significant word of ”X” (0x12345678) (in Little Endian Order)**

**0x10000100 – 0x78**

**0x10000101 – 0x56**

**0x10000102 – 0x34**

**0x10000103 – 0x12**

**Update the memory 0x10000104 with the Most significant word of ”X” (0x12345678) (in Little Endian Order)**

**0x10000104 – 0x78**

**0x10000105 – 0x56**

**0x10000106 – 0x34**

**0x10000107 – 0x12**

**Update the memory 0x10000108 with the Least significant word of ”Y” (0x23456789) (in Little Endian Order)**

**0x10000108 – 0x89**

**0x10000109 – 0x67**

**0x1000010A – 0x45**

**0x1000010B – 0x23**

**Update the memory 0x1000010C with the Most significant word of ”Y” (0x23456789) (in Little Endian Order)**

**0x1000010C – 0x78**

**0x1000010D – 0x56**

**0x1000010E – 0x34**

**0x1000010F – 0x12**

1. g

**Assembly Program:**

**AREA mydata, DATA**

**X0 EQU 0x10000100**

**X1 EQU 0x10000104**

**Y0 EQU 0x10000108**

**Y1 EQU 0x1000010C**

**RE0 EQU 0x10000110**

**RE1 EQU 0x10000114**

**AREA gpio, CODE, READONLY, ALIGN=2**

**EXPORT \_\_main**

**ENTRY**

**\_\_main**

**LDR R0, =X0 ;Get the address of Least significant word(X0)'s address**

**LDR R1, [R0]**

**LDR R0, =X1 ;Get the address of Most significant word(X1)'s address**

**LDR R2, [R0]**

**LDR R0, =Y0 ;Get the address of Least significant word(Y0)'s address**

**LDR R3, [R0]**

**LDR R0, =Y1 ;Get the address of Most significant word(Y1)'s address**

**LDR R4, [R0]**

**ADDS R1, R1,R3**

**ADC R2, R2, R4**

**LDR R0, =RE0**

**STR R1, [R0]**

**LDR R0, =RE1**

**STR R2, [R0]**

**LBL B LBL**

**END**

**Output:**

Figure 1. Watch window showing PORTB value as 0x01

A screenshot of a computer

Description automatically generated

Figure 1. Memory Window View

**Work for Students: (To be submitted on or before 29/10/21)**

1. Modify the program to include carry.
2. Modify the program to do subtraction of two 64 bit numbers with borrow.
3. Write a program to Multiply two 64 bit numbers and get the 64 bit results

**Conclusion:** Thus, an assembly program is written to add two 64 bit numbers in memory and put the result back to memory.

Project folder: Arithmetic

**Exercise 2:** Write a ARM assembly program to fill the section of memory with particular byte and then transfer the block of memory to another location

**Objective**: To write an assembly language program to fill the section of memory with 0xAA and then transfer the block of memory (100 bytes) to another location. And the program will be simulated.

**Algorithm:**

**Assembly Program:**

AREA mydata, DATA

FILL\_VAL EQU 0xAAAAAAAA

SOURCE EQU 0x10000100

DEST EQU 0x10000300

COUNT EQU 100

AREA MEM\_OP, CODE, READONLY, ALIGN=2

EXPORT \_\_main

ENTRY

\_\_main

LDR R0, =SOURCE

LDR R1, =FILL\_VAL

LDR R2, =COUNT

LDR R3,=DEST

;FILL THE MEMORY WITH 0xAA

STO STR R1,[R0], #4

SUBS R2,R2,#4

BNE STO

;RESTORE THE REGISTER VALUES

LDR R0, =SOURCE

LDR R2, =COUNT

;MOVE THE SOURCE TO DEST

STO1 LDR R4,[R0],#4

STR R4, [R3],#4

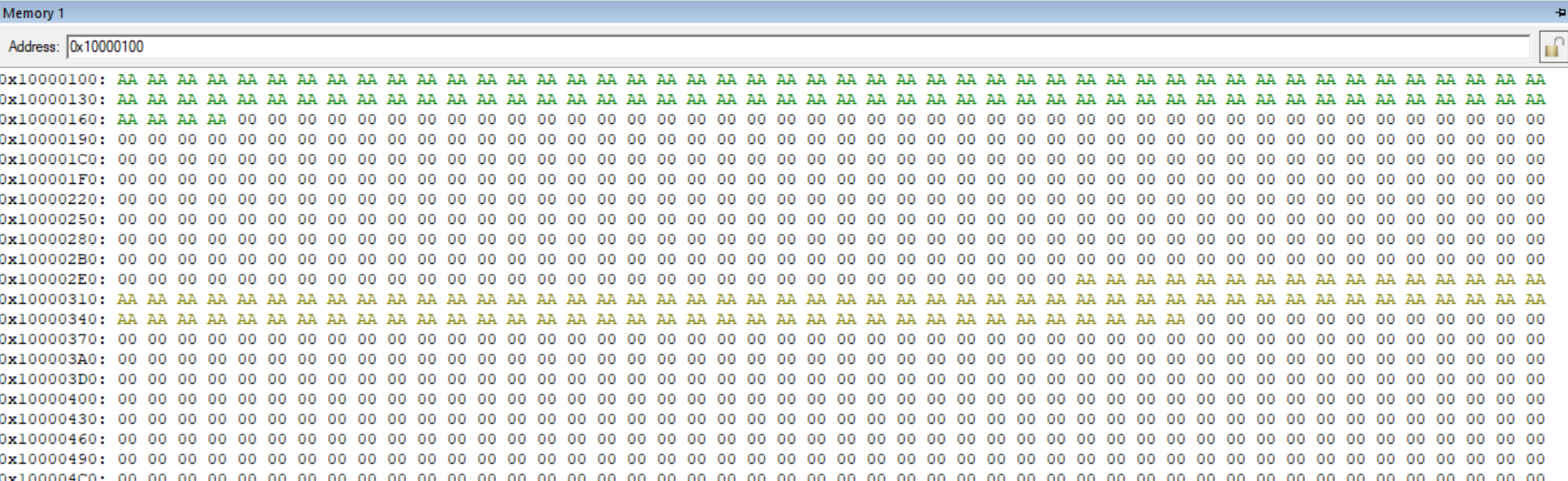
SUBS R2,R2,#4

BNE STO1

L B L

END

Output:



**Conclusion:** Thus, an assembly program is written to fill the memory block with particular byte pattern and copy the block of memory to another location.

Project folder: mem\_op

**Exercise 3:** Write an ARM C program to Blink the LEDs connected to General Purpose Input/Output pins

**Objective**: To write an C language program to Blink the LEDs connected to General Purpose Input/Output pins. And the program will be simulated.

**Algorithm:**

1. Make

**C Program:**

#include <stdio.h>

#include "LPC17xx.H" /\* LPC17xx definitions \*/

/\*----------------------------------------------------------------------------

Main Program

\*----------------------------------------------------------------------------\*/

int main (void) {

uint32\_t i,j;

LPC\_GPIO1->FIODIR |= (1<<23); /\* Buzzer on PORT1.23 are output \*/

LPC\_GPIO1->FIODIR |= (1<<22); /\* Relay on PORT1.22 are output \*/

LPC\_GPIO0->FIODIR |= (1<<0)| (1<<1) | (1<< 10) | (1<<11); // 1111 1111 1111 1111 1111 1111 1100 0011

LPC\_GPIO1->FIOPIN &= ~(1<<23); // To switch off the buzzer at P1.23

while(1)

{

LPC\_GPIO0->FIOPIN |= (1<<0)| (1<<1) | (1<< 10) | (1<<11);

LPC\_GPIO1->FIOPIN |= (1<<22); // P1.22 (Relay)

for(i=0;i<300;i++)

{

for (j=0;j<100000;j++);

}

LPC\_GPIO0->FIOPIN &= ~((1<<0)| (1<<1) | (1<< 10) | (1<<11));

LPC\_GPIO1->FIOPIN &= ~(1<<22);

for(i=0;i<300;i++)

{

for (j=0;j<100000;j++);

}

}

}

Output:

**Board used:** Innowitech Solutions LPC1768 Board

**Conclusion:** Thus, a C program is written to Toggle the LEDs connected to P1.0, P1.1, P1.10, P1.11 and the relay connected to P1.22. And the program also switches off the Buzzer (P1.23).

Project folder: Blinky1

**Exercise 4:** Write a ARM C program to demonstrate the external interrupt

**Objective**: To write a C program to generate an interrupt when the button connected to EINT3 pin is pressed and activate the relay from the ISR.

**Algorithm:**

**C Program:**

#include "LPC17xx.H" /\* LPC17xx definitions \*/

/\*----------------------------------------------------------------------------

Function that initializes Button INT1 to generate an interrupt

\*----------------------------------------------------------------------------\*/

void BUTTON\_Init(void) {

LPC\_GPIO2->FIODIR &= ~(1 << 11); /\* PORT2.11 defined as input \*/

LPC\_GPIOINT->IO2IntEnF |= (1 << 11); /\* enable falling edge irq \*/

NVIC\_EnableIRQ(EINT3\_IRQn); /\* enable irq in nvic \*/

}

/\*----------------------------------------------------------------------------

External IRQ Handler

\*---------------------------------------------------------------------------\*/

void EINT3\_IRQHandler()

{

static uint8\_t RelayStatus = 0;

LPC\_GPIOINT->IO2IntClr |= (1 << 11); /\* clear pending interrupt \*/

if (RelayStatus == 0)

{

LPC\_GPIO1->FIOPIN |= (1<<22);

RelayStatus = 1;

}

else

{

LPC\_GPIO1->FIOPIN &= ~(1<<22);

RelayStatus = 0;

}

}

/\*----------------------------------------------------------------------------

Main Program

\*----------------------------------------------------------------------------\*/

int main (void) {

// LED\_Init();

LPC\_GPIO1->FIODIR |= (1 << 22); /\* Relay pin, PORT1.22 defined as output \*/

LPC\_GPIO1->FIODIR |= (1 << 23); /\* Buzzer pin, PORT1.23 defined as output \*/

LPC\_GPIO1->FIOPIN &= ~(1<<23); // Stop the buzzer

BUTTON\_Init();

while (1);

}

Output:

**Conclusion:** Thus, an C program has been developed for EINT3 interrupt and the relay is toggled from the ISR.

Project folder: EXTI\_1

**Exercise 5:** Write a C program to demonstrate the UART transmission and reception

**Objective**: To write an C program to get a character from the serial port and print string in the serial console.

**Algorithm:**

**Assembly Program:**

/\*----------------------------------------------------------------------------

\* Name: Blinky.c

\* Purpose: LED Flasher

\* Note(s): possible defines set in "options for target - C/C++ - Define"

\* \_\_USE\_LCD - enable Output on LCD

\*----------------------------------------------------------------------------

\* This file is part of the uVision/ARM development tools.

\* This software may only be used under the terms of a valid, current,

\* end user licence from KEIL for a compatible version of KEIL software

\* development tools. Nothing else gives you the right to use this software.

\*

\* This software is supplied "AS IS" without warranties of any kind.

\*

\* Copyright (c) 2008-2011 Keil - An ARM Company. All rights reserved.

\*----------------------------------------------------------------------------\*/

#include <stdio.h>

#include "LPC17xx.H" /\* LPC17xx definitions \*/

#define \_\_FI 1 /\* Font index 16x24 \*/

void Serial\_Init (void);

int Serial\_PutChar (int c);

int Serial\_GetChar (void);

/\*----------------------------------------------------------------------------

Main Program

\*----------------------------------------------------------------------------\*/

int main (void) {

uint32\_t i, j;

uint8\_t ch;

Serial\_Init(); /\* UART Initialization \*/

LPC\_GPIO1->FIODIR |= (1 << 23); /\* Buzzer pin, PORT1.22 defined as output \*/

LPC\_GPIO1->FIOPIN &= ~(1<<23); // Stop the buzzer

while (1) { /\* Loop forever \*/

Serial\_PutChar('H');

Serial\_PutChar('E');

Serial\_PutChar('L');

Serial\_PutChar('L');

Serial\_PutChar('O');

Serial\_PutChar('\r');

Serial\_PutChar('\n');

ch = Serial\_GetChar();

Serial\_PutChar(ch);

Serial\_PutChar('\r');

Serial\_PutChar('\n');

// delay

for (j=0;j<100000;j++);

}

}

void Serial\_Init (void) {

LPC\_SC->PCONP |= ((1 << 3) | (1 << 15)); /\* enable power to UART0 & IOCON \*/

LPC\_PINCON->PINSEL0 |= (1 << 4); /\* Pin P0.2 used as TXD0 \*/

LPC\_PINCON->PINSEL0 |= (1 << 6); /\* Pin P0.3 used as RXD0 \*/

LPC\_UART0->LCR = 0x83; /\* 8 bits, no Parity, 1 Stop bit \*/

LPC\_UART0->DLL = 9; /\* 115200 Baud Rate @ 25.0 MHZ PCLK\*/

LPC\_UART0->FDR = 0x21; /\* FR 1,507, DIVADDVAL=1, MULVAL=2 \*/

LPC\_UART0->DLM = 0; /\* High divisor latch = 0 \*/

LPC\_UART0->LCR = 0x03; /\* DLAB = 0 \*/

}

/\*----------------------------------------------------------------------------

Write character to Serial Port

\*----------------------------------------------------------------------------\*/

int Serial\_PutChar (int c) {

while (!(LPC\_UART0->LSR & 0x20));

LPC\_UART0->THR = c;

return (c);

}

/\*----------------------------------------------------------------------------

Read character from Serial Port (blocking read)

\*----------------------------------------------------------------------------\*/

int Serial\_GetChar (void) {

while (!(LPC\_UART0->LSR & 0x01));

return (LPC\_UART0->RBR);

}

Output:

**Conclusion:** Thus, a C program has been developed for the UART transmission and reception.

Project folder: UART

**Exercise 6:** Write a C program to read the digital equivalent of analog value given at the analog channel.

**Objective**: To write a C program to print the digital equivalent of analog value given at the analog channel to the serial console.

**Algorithm:**

**Assembly Program:**

#include <stdio.h>

#include "LPC17xx.H" /\* LPC17xx definitions \*/

void Serial\_Init (void);

int Serial\_PutChar (int );

void ADC\_Initialize (void);

void ADC\_StartConv (void);

char text[40];

#define ADC\_VALUE\_MAX (0xFFF)

uint16\_t AD\_last;

uint8\_t AD\_done;

/\*----------------------------------------------------------------------------

Main Program

\*----------------------------------------------------------------------------\*/

int main (void) {

uint32\_t i, j;

Serial\_Init(); /\* UART Initialization \*/

ADC\_Initialize(); /\* ADC Initialization \*/

LPC\_GPIO1->FIODIR |= (1 << 23); /\* Buzzer pin, PORT1.22 defined as output \*/

LPC\_GPIO1->FIOPIN &= ~(1<<23); // Stop the buzzer

while (1) { /\* Loop forever \*/

ADC\_StartConv ();

/\* AD converter input \*/

if (AD\_done) { /\* If conversion has finished \*/

AD\_done = 0;

sprintf(text, "AD value: 0x%04X", AD\_last); /\* format text for print out \*/

for (i=0;text[i] != 0; i++)

Serial\_PutChar(text[i]);

Serial\_PutChar('\r');

Serial\_PutChar('\n');

}

for (j=0;j<100000; j++);

}

}

void Serial\_Init (void) {

LPC\_SC->PCONP |= ((1 << 3) | (1 << 15)); /\* enable power to UART0 & IOCON \*/

LPC\_PINCON->PINSEL0 |= (1 << 4); /\* Pin P0.2 used as TXD0 \*/

LPC\_PINCON->PINSEL0 |= (1 << 6); /\* Pin P0.3 used as RXD0 \*/

LPC\_UART0->LCR = 0x83; /\* 8 bits, no Parity, 1 Stop bit \*/

LPC\_UART0->DLL = 9; /\* 115200 Baud Rate @ 25.0 MHZ PCLK\*/

LPC\_UART0->FDR = 0x21; /\* FR 1,507, DIVADDVAL=1, MULVAL=2 \*/

LPC\_UART0->DLM = 0; /\* High divisor latch = 0 \*/

LPC\_UART0->LCR = 0x03; /\* DLAB = 0 \*/

}

void ADC\_Initialize (void) {

LPC\_SC->PCONP |= ((1 << 12) | (1 << 15)); /\* enable power to ADC & IOCON \*/

LPC\_PINCON->PINSEL3 &= ~( 3 << 30);

LPC\_PINCON->PINSEL3 |= ( 3 << 30); /\* P0.25 is AD0.2 \*/

LPC\_PINCON->PINMODE3 &= ~( 3 << 30);

LPC\_PINCON->PINMODE3 |= ( 3 << 30); /\* P0.25 no pull up/down \*/

LPC\_ADC->ADCR = ( 1 << 5) | /\* select AD0.5 pin \*/

( 4 << 8) | /\* ADC clock is 25MHz/5 \*/

( 1 << 21); /\* enable ADC \*/

#ifdef \_\_ADC\_IRQ

LPC\_ADC->ADINTEN = ( 1 << 8); /\* global enable interrupt \*/

NVIC\_EnableIRQ(ADC\_IRQn); /\* enable ADC Interrupt \*/

#endif

}

/\*----------------------------------------------------------------------------

start AD Conversion

\*----------------------------------------------------------------------------\*/

void ADC\_StartConv (void) {

LPC\_ADC->ADCR &= ~( 7 << 24); /\* stop conversion \*/

LPC\_ADC->ADCR |= ( 1 << 24); /\* start conversion \*/

}

/\*----------------------------------------------------------------------------

A/D IRQ: Executed when A/D Conversion is done

\*----------------------------------------------------------------------------\*/

#ifdef \_\_ADC\_IRQ

void ADC\_IRQHandler(void) {

volatile uint32\_t adstat;

adstat = LPC\_ADC->ADSTAT; /\* Read ADC clears interrupt \*/

AD\_last = (LPC\_ADC->ADGDR >> 4) & ADC\_VALUE\_MAX; /\* Store converted value \*/

AD\_done = 1;

}

#endif

/\*----------------------------------------------------------------------------

Write character to Serial Port

\*----------------------------------------------------------------------------\*/

int Serial\_PutChar (int c) {

while (!(LPC\_UART0->LSR & 0x20));

LPC\_UART0->THR = c;

return (c);

}

Output:

**Conclusion:** Thus, a C program is written to print the digital equivalent of analog value given at the analog channel to the serial console.

Project folder: ADC

**Exercise 7:** Write a C program to Configure the RTC and get the date and time from RTC periodically.

**Objective**: To write a C program to Configure the RTC with the date and time and to get the date and time from RTC for every 1 second.

**Algorithm:**

**C Program:**

#include<lpc17xx.h>

#include <stdio.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Commonly used RTC macros/Constants

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define SBIT\_CLKEN 0 /\* RTC Clock Enable\*/

#define SBIT\_CTCRST 1 /\* RTC Clock Reset \*/

#define SBIT\_CCALEN 4 /\* RTC Calibration counter enable \*/

#define SBIT\_CALDIR 17

#define SBIT\_PCRTC 9

typedef struct

{

uint8\_t sec;

uint8\_t min;

uint8\_t hour;

uint8\_t weekDay;

uint8\_t date;

uint8\_t month;

uint16\_t year;

}rtc\_t;

char text[50];

void RTC\_Init(void);

void RTC\_SetDateTime(rtc\_t \*);

void RTC\_GetDateTime(rtc\_t \*);

void Serial\_Init (void);

void DELAY\_us(unsigned int count);

int Serial\_PutChar (int c);

int main()

{

rtc\_t rtc;

int i;

SystemInit();

Serial\_Init () ;

//buzzer off

LPC\_GPIO1->FIODIR |=(1<<23);

LPC\_GPIO1->FIOCLR |=(1<<23); //P1.23 low buzzer

RTC\_Init();

rtc.hour = 13; // 10:40:20 am

rtc.min = 40;

rtc.sec = 0;

rtc.date = 24; //1st Jan 2016

rtc.month = 12;

rtc.year = 21;

rtc.weekDay = 5; // Friday: 5th day of week considering monday as first day.

/\*##### Set the time and Date only once. Once the Time and Date is set, comment these lines

and reflash the code. Else the time will be set every time the controller is reset\*/

RTC\_SetDateTime(&rtc); // 10:40:20 am, 1st Jan 2016

/\* Display the Time and Date continuously \*/

while(1)

{

RTC\_GetDateTime(&rtc);

sprintf(text, "time:%2d:%2d:%2d Date:%2d/%2d/%2d\r\n",(uint16\_t)rtc.hour,(uint16\_t)rtc.min,(uint16\_t)rtc.sec,(uint16\_t)rtc.date,(uint16\_t)rtc.month,(uint16\_t)rtc.year);

for(i = 0; text[i] != '\0'; i++)

Serial\_PutChar (text[i]);

DELAY\_us(1000000);

}

}

void RTC\_Init(void)

{

LPC\_SC->PCONP |=(1<<9); // Enable the power to RTC module

/\* Disable RTC clock, reset clock, Stops RTC calibration \*/

LPC\_RTC->CCR = ((1<<SBIT\_CTCRST) | (1<<SBIT\_CCALEN));

LPC\_RTC->CALIBRATION = 0x00;

LPC\_RTC->CCR = (1<<SBIT\_CLKEN); /\* Enable the clock for RTC \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void RTC\_SetDateTime(rtc\_t \*rtc)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* I/P Arguments: rtc\_t \*: Pointer to structure of type rtc\_t. Structure contains hour,min,sec,day,date,month and year

\* Return value : none

\* description :This function is used to set Date(dd,mm,yy) into the RTC.

The new Date is updated into the non volatile memory of RTC .

Note: The I/P arguments should of hex or decimal

like 15,08,2047 for 15th day,8th month and 2047th year.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void RTC\_SetDateTime(rtc\_t \*rtc)

{

LPC\_RTC->SEC = rtc->sec; // Update sec value

LPC\_RTC->MIN = rtc->min; // Update min value

LPC\_RTC->HOUR = rtc->hour; // Update hour value

LPC\_RTC->DOW = rtc->weekDay; // Update day value

LPC\_RTC->DOM = rtc->date; // Update date value

LPC\_RTC->MONTH = rtc->month; // Update month value

LPC\_RTC->YEAR = rtc->year; // Update year value

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void RTC\_GetDateTime(rtc\_t \*rtc)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* I/P Arguments: rtc\_t \*: Pointer to structure of type rtc\_t. Structure contains hour,min,sec,day,date,month and year

\* Return value : none

\* description :This function is used to get the Time(hh,mm,ss) And Date from RTC.

Note: The time read from will be of hex or decimal,

like 12,39,26 for 12hr,39min and 26sec.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void RTC\_GetDateTime(rtc\_t \*rtc)

{

rtc->sec = LPC\_RTC->SEC ; // Read sec value

rtc->min = LPC\_RTC->MIN ; // Read min value

rtc->hour = LPC\_RTC->HOUR; // Read hour value

rtc->weekDay = LPC\_RTC->DOW; // Read day value

rtc->date = LPC\_RTC->DOM; // Read date value

rtc->month = LPC\_RTC->MONTH; // Read month value

rtc->year = LPC\_RTC->YEAR; // Read year value

}

void Serial\_Init (void) {

LPC\_SC->PCONP |= ((1 << 3) | (1 << 15)); /\* enable power to UART0 & IOCON \*/

LPC\_PINCON->PINSEL0 |= (1 << 4); /\* Pin P0.2 used as TXD0 \*/

LPC\_PINCON->PINSEL0 |= (1 << 6); /\* Pin P0.3 used as RXD0 \*/

LPC\_UART0->LCR = 0x83; /\* 8 bits, no Parity, 1 Stop bit \*/

LPC\_UART0->DLL = 9; /\* 115200 Baud Rate @ 25.0 MHZ PCLK\*/

LPC\_UART0->FDR = 0x21; /\* FR 1,507, DIVADDVAL=1, MULVAL=2 \*/

LPC\_UART0->DLM = 0; /\* High divisor latch = 0 \*/

LPC\_UART0->LCR = 0x03; /\* DLAB = 0 \*/

}

void DELAY\_us(unsigned int count)

{

unsigned int j=0,i=0;

for(j=0;j<count;j++)

{

/\* At 100Mhz, the below loop introduces

DELAY of 1 us \*/

for(i=0;i<19;i++); //23

}

}

/\*----------------------------------------------------------------------------

Write character to Serial Port

\*----------------------------------------------------------------------------\*/

int Serial\_PutChar (int c) {

while (!(LPC\_UART0->LSR & 0x20));

LPC\_UART0->THR = c;

return (c);

}

Output:

**Conclusion:** Thus, an assembly program is written to fill the memory block with particular byte pattern and copy the block of memory to another location.

Project folder: RTC

**Exercise 8:** Write a FreeRTOS C program to demonstrate the task scheduling

**Objective**: To write a FreeRTOS C program to create two tasks and demonstrate the task scheduling and switching.

**Algorithm:**

**C Program:**

/\* Scheduler include files. \*/

#include "FreeRtOSConfig.h"

#include "FreeRTOS.h"

#include "task.h"

#include "croutine.h"

#include "uart.h" //Explore Embedded UART library

/\* Local Tasks declaration \*/

static void MyTask1(void\* pvParameters);

static void MyTask2(void\* pvParameters);

#define LED\_Task1 0x02u

#define LED\_Task2 0x04u

#define LED\_PORT LPC\_GPIO2->FIOPIN

int main(void)

{

SystemInit(); /\* Initialize the controller \*/

UART\_Init(38400); /\* Initialize the Uart module \*/

LPC\_GPIO2->FIODIR = 0xffffffffu;

/\* Create the three tasks with priorities 1,2,3. Only tasks will be created.

\* Tasks will be excecuted once the scheduler is started.

\* An idle task is also created, which will be run when there are no tasks in RUN state \*/

xTaskCreate( MyTask1, ( signed char \* )"Task1", configMINIMAL\_STACK\_SIZE, NULL, 1, NULL );

xTaskCreate( MyTask2, ( signed char \* )"Task2", configMINIMAL\_STACK\_SIZE, NULL, 2, NULL );

UART\_Printf("\n\rIn the main");

vTaskStartScheduler(); /\* Start the schedular \*/

while(1);

}

/\* Task1 with priority 1 \*/

static void MyTask1(void\* pvParameters)

{

unsigned int i;

while(1)

{

LED\_PORT = LED\_Task1; /\* Led to indicate the execution of Task1\*/

UART\_Printf("\n\rTask1"); //reentrant

//uxTaskPrioritySet(

// change task1 priority to 3

for (i =0;i< 0xffff;i++); // Lower priority task can do continuous processing

}

}

/\* Task2 with priority 2 \*/

static void MyTask2(void\* pvParameters)

{

unsigned int i;

while(1)

{

LED\_PORT = LED\_Task2; /\* Led to indicate the execution of Task2\*/

UART\_Printf("\n\rTask2");

vTaskDelay(150);

}

}

Output:

**Conclusion:** Thus, a FreeRTOS C program to demonstrate the task scheduling and switching.

Project folder: FreeRTOS-Task

**Exercise 9:** Write a FreeRTOS C program to demonstrate the inter task communication

**Objective**: To write a FreeRTOS C program to demonstrate the inter task communication

**Algorithm:**

**C Program:**

/\* Scheduler include files. \*/

#include "FreeRtOSConfig.h"

#include "FreeRTOS.h"

#include "task.h"

#include "queue.h"

#include "croutine.h"

#include "uart.h" //Explore Embedded UART library

/\* Local Tasks declaration \*/

static void MyTask1(void\* pvParameters);

static void MyTask2(void\* pvParameters);

#define LED\_Task1 0x02u

#define LED\_Task2 0x04u

#define LED\_PORT LPC\_GPIO2->FIOPIN

/\* Declare a variable of type QueueHandle\_t. This is used to store the queue

that is accessed by all three tasks. \*/

xQueueHandle xQueue;

int main(void)

{

SystemInit(); /\* Initialize the controller \*/

UART\_Init(38400); /\* Initialize the Uart module \*/

LPC\_GPIO2->FIODIR = 0xffffffffu;

/\* The queue is created to hold a maximum of 5 long values. \*/

xQueue = xQueueCreate( 5, sizeof( int32\_t ) );

if( xQueue != NULL )

{

/\* Create the three tasks with priorities 1,2,3. Only tasks will be created.

\* Tasks will be excecuted once the scheduler is started.

\* An idle task is also created, which will be run when there are no tasks in RUN state \*/

xTaskCreate( MyTask1, ( signed char \* )"Task1", configMINIMAL\_STACK\_SIZE, NULL, 1, NULL );

xTaskCreate( MyTask2, ( signed char \* )"Task2", configMINIMAL\_STACK\_SIZE, NULL, 2, NULL );

UART\_Printf("\n\rIn the main");

vTaskStartScheduler(); /\* Start the schedular \*/

}

while(1);

}

/\* Task1 with priority 1 \*/

static void MyTask1(void\* pvParameters)

{

unsigned int i;

int32\_t lValueToSend = 0x1234;

portBASE\_TYPE xStatus;

while(1)

{

LED\_PORT = LED\_Task1; /\* Led to indicate the execution of Task1\*/

UART\_Printf("\n\rTask1 tries to send the message to message queue"); //reentrant

//uxTaskPrioritySet(

// change task1 priority to 3

xStatus = xQueueSendToBack( xQueue, &lValueToSend, 0 );

if( xStatus != pdPASS )

{

/\* We could not write to the queue because it was full – this must

be an error as the queue should never contain more than one item! \*/

UART\_Printf("Could not send to the queue.\r\n" );

}

else

{

UART\_Printf("\n\rTask1 sent the message");

}

vTaskDelay(150);

//for (i =0;i< 0xffff;i++); // Lower priority task can do continuous processing

}

}

/\* Task2 with priority 2 \*/

static void MyTask2(void\* pvParameters)

{

unsigned int i;

int32\_t lReceivedValue;

portBASE\_TYPE xStatus;

while(1)

{

LED\_PORT = LED\_Task2; /\* Led to indicate the execution of Task2\*/

// UART\_Printf("\n\rTask2");

UART\_Printf("\n\rTask2 tries to receive the message\n");

xStatus = xQueueReceive( xQueue, &lReceivedValue, 100 ); //portMAX\_DELAY

if( xStatus == pdPASS )

{

/\* Data was successfully received from the queue, print out the received

value. \*/

UART\_Printf(" Data Received = 0x%x\n", lReceivedValue );

}

else

{

/\* We did not receive anything from the queue even after waiting for 100ms.

This must be an error as the sending tasks are free running and will be

continuously writing to the queue. \*/

UART\_Printf("Could not receive from the queue.\r\n" );

}

//vTaskDelay(150);

//for (i =0;i< 0xfffff;i++);

}

}

Output:

**Conclusion:** Thus, a FreeRTOS C program has been developed to demonstrate the inter task communication.

Project folder: FreeRTOS-Task-Comm