**MEMORANDUM**

**To:** Dr. Thornburg

**From:** Ryan Wise and Paul Hertfelder

**Date:** 12/04/2017

**Subject: Lab 6 RTOS Report**

**Introduction**

The purpose of this lab is to design and implement a program that solves a computational task while utilizing µC/OS-III. The program must utilize key aspects of RTOS including at least 3 tasks and preferably additional semaphores, priority levels, message queues, etc.

**Design**

The design of this open lab was designed with a calculator in mind. This calculator uses µC/OS-III and allows a user to input an item purchased by pressing a button on the PIC32 board. This item has an associated cost, and this cost is added to a total cost. The number of times the buttons are pressed is stored to display the quantity of items sold. Three of the four buttons are used with each three representing a different item. A toggle switch is also used on the PIC32 board which switches between displaying the quantity of items sold and the total cost. The 8 LEDs are used on the PIC32 board to display the binary representation of the selected output data. A flowchart was made to show the overall design of the program which is shown in Figure 1.

The code consists of three tasks, a polling task, a counting task, and an LED task. The polling checks the state of the three buttons and the switch. If any buttons are pressed, the task sends the respective value of the given item to the counting task via a message queue. The counting task increments the total number of items and adds the message value to the cost of items sold. Once complete, the counting task posts a semaphore to update the LEDs with the new values. The respective priorities of these tasks consist of the counting task being priority 2, the polling task being priority 3, and the LED task being priority 4.

**Testing & Result**

Testing was done by doing hand calculations with each press of an item button. The first test was simply to check to see, that for each button press, the LEDs would light up to display the correct quantity. This was proven to be working as each button press between the three buttons used would increase the quantity by one. The second test that was done was to check to see if the different items with a different cost would add up to the correct total amount. The results are shown in table 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Items Sold | Cost (Decimal) | Cost (Binary) | Quantity (Decimal) | Quantity (Binary) |
| 1(1$)+2(3$)+1(4$) | 11 | 0000 1011 | 4 | 0000 0100 |
| 2(1$)+5(3$) | 17 | 0001 0001 | 7 | 0000 0111 |
| 30(4$) | 120 | 0111 1000 | 30 | 0001 1110 |
| 10(1$)+10(3$)+4(4$) | 56 | 0011 1000 | 24 | 0001 1000 |

**Table 1: Cost\Quantity Outputs**

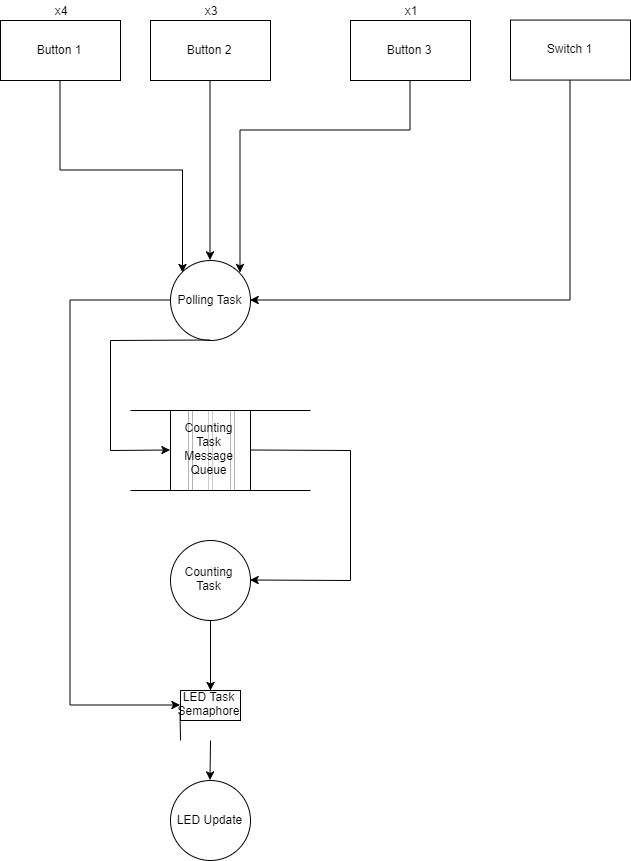
|  |  |  |  |
| --- | --- | --- | --- |
| Properties | Counting | Polling | LED |
| CPU Usage | 0% | 0% | 0% |
| Stack Size | 512 | 512 | 512 |
| Words Used | 67 | 67 | 63 |
| Words Free | 445 | 445 | 449 |
| Overall CPU Usage | .01% | .01% | .01% |
| Total Memory Used | 13% | 13% | 13% |

**Table 2: Task Utilization**

**Conclusion**

The lab brought up multiple ideas that were presented in class. The successfulness of the lab depended on multiple tasks and a message queue. An issue that occurred was in trying to use the ISR to interrupt when a user pushes a button or uses the toggle switch rather than polling these same items. Ideally, the hardware interrupt would allow for a hardware debounce circuit to be used as well as allowed the user to press the button as quickly and often as liked. Unfortunately, despite checking the datasheet for the location of the hardware interrupt pins, they appeared unresponsive. Due to time constraints, button reading was switched to polling. Polling has the drawback of not being debounced (though, given more time, this could have been rectified), requiring CPU time to check on the state of these buttons often, and, with our implementation, holding the buttons down for too long registers as multiple presses. Other than these small details, the lab functions as planned.

**Appendix:**

****

**Figure 1: RTOS Code Flowchart**

**#include <includes.h>**

**#include <stdio.h>**

**#include <stdarg.h>**

**#include <stddef.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <ctype.h>**

**/\***

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

**\* VARIABLES**

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

**\*/**

**static OS\_TCB App\_TaskStartTCB;**

**static OS\_TCB PollingTCB;**

**static OS\_TCB CountingTCB;**

**static OS\_TCB LED\_TCB;**

**static CPU\_STK App\_TaskStartStk[APP\_CFG\_TASK\_START\_STK\_SIZE];**

**static CPU\_STK PollingStk[APP\_CFG\_TASK\_START\_STK\_SIZE];**

**static CPU\_STK LEDStk[APP\_CFG\_TASK\_START\_STK\_SIZE];**

**static CPU\_STK CountingStk[APP\_CFG\_TASK\_START\_STK\_SIZE];**

**CPU\_INT08U j;**

**static CPU\_INT08U toggle\_value;**

**CPU\_INT08U costs[3] = {4, 3, 1};**

**/\***

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

**\* FUNCTION PROTOTYPES**

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

**\*/**

**static void App\_TaskCreate (void);**

**static void App\_ObjCreate (void);**

**static void Polling (void);**

**static void LED (void);**

**static void Counting (void);**

**static void App\_TaskStart (void \*p\_arg);**

**/\***

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

**\* main()**

**\***

**\* Description : This is the standard entry point for C code.**

**\***

**\* Arguments : none**

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

**\*/**

**OS\_ERR os\_err;**

**CPU\_TS ts;**

**OS\_MUTEX Multiplier\_Mutex;**

**static CPU\_INT16U bottles;**

**static CPU\_INT16U cans;**

**static CPU\_INT08U sodas;**

**static CPU\_INT32U sales;**

**static CPU\_INT32U quantity;**

**int main (void)**

**{**

**ConfigINT0(EXT\_INT\_PRI\_0|RISING\_EDGE\_INT|EXT\_INT\_ENABLE);**

**ConfigINT1(EXT\_INT\_PRI\_1|RISING\_EDGE\_INT|EXT\_INT\_ENABLE);**

**ConfigINT2(EXT\_INT\_PRI\_2|RISING\_EDGE\_INT|EXT\_INT\_ENABLE);**

**CPU\_Init(); /\* Initialize the uC/CPU services \*/**

**BSP\_IntDisAll();**

**OSInit(&os\_err); /\* Init uC/OS-III. \*/**

**OSTaskCreate((OS\_TCB \*)&App\_TaskStartTCB, /\* Create the start task \*/**

**(CPU\_CHAR \*)"Start",**

**(OS\_TASK\_PTR )App\_TaskStart,**

**(void \*)0,**

**(OS\_PRIO )APP\_CFG\_TASK\_START\_PRIO,**

**(CPU\_STK \*)&App\_TaskStartStk[0],**

**(CPU\_STK\_SIZE )APP\_CFG\_TASK\_START\_STK\_SIZE\_LIMIT,**

**(CPU\_STK\_SIZE )APP\_CFG\_TASK\_START\_STK\_SIZE,**

**(OS\_MSG\_QTY )0u,**

**(OS\_TICK )0u,**

**(void \*)0,**

**(OS\_OPT )(OS\_OPT\_TASK\_STK\_CHK | OS\_OPT\_TASK\_STK\_CLR),**

**(OS\_ERR \*)&os\_err);**

**OSStart(&os\_err); /\* Start multitasking (i.e. give control to uC/OS-III). \*/**

**(void)&os\_err;**

**return (0);**

**}**

**/\*$PAGE\*/**

**/\***

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

**\* STARTUP TASK**

**\***

**\* Description : This is an example of a startup task. As mentioned in the book's text, you MUST**

**\* initialize the ticker only once multitasking has started.**

**\* Arguments : p\_arg is the argument passed to 'AppStartTask()' by 'OSTaskCreate()'.**

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

**\*/**

**static void App\_TaskStart (void \*p\_arg)**

**{**

**(void)p\_arg;**

**BSP\_InitIO(); /\* Initialize BSP functions \*/**

**Mem\_Init(); /\* Initialize memory managment module \*/**

**Math\_Init(); /\* Initialize mathematical module \*/**

**#if (OS\_CFG\_STAT\_TASK\_EN > 0u)**

**OSStatTaskCPUUsageInit(&os\_err); /\* Determine CPU capacity \*/**

**#endif**

**#ifdef CPU\_CFG\_INT\_DIS\_MEAS\_EN**

**CPU\_IntDisMeasMaxCurReset();**

**#endif**

**App\_TaskCreate(); /\* Create Application tasks \*/**

**App\_ObjCreate(); /\* Create Application kernel objects \*/**

**OSTaskSemPend(0, OS\_OPT\_PEND\_BLOCKING, &ts ,&os\_err);**

**}**

**/\***

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

**\* AppTaskCreate()**

**\***

**\* Description : Create application tasks.**

**\***

**\* Argument(s) : none**

**\***

**\* Return(s) : none**

**\***

**\* Caller(s) : AppTaskStart()**

**\***

**\* Note(s) : none.**

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

**\*/**

**static void App\_TaskCreate (void)**

**{**

**OSMutexCreate (&Multiplier\_Mutex, "Multipler Mutex", &os\_err);**

**OSMutexPost (&Multiplier\_Mutex, OS\_OPT\_PEND\_BLOCKING, &os\_err);**

**OSTaskCreate((OS\_TCB \*)&PollingTCB, /\* Create the start task \*/**

**(CPU\_CHAR \*)"Polling Task",**

**(OS\_TASK\_PTR )Polling,**

**(void \*)0,**

**(OS\_PRIO )APP\_CFG\_POLLING\_PRIO,**

**(CPU\_STK \*)&PollingStk[0],**

**(CPU\_STK\_SIZE )APP\_CFG\_TASK\_START\_STK\_SIZE\_LIMIT,**

**(CPU\_STK\_SIZE )APP\_CFG\_TASK\_START\_STK\_SIZE,**

**(OS\_MSG\_QTY )0u,**

**(OS\_TICK )0u,**

**(void \*)0,**

**(OS\_OPT )(OS\_OPT\_TASK\_STK\_CHK | OS\_OPT\_TASK\_STK\_CLR),**

**(OS\_ERR \*)&os\_err);**

**OSTaskCreate((OS\_TCB \*)&LED\_TCB, /\* Create the start task \*/**

**(CPU\_CHAR \*)"LED",**

**(OS\_TASK\_PTR )LED,**

**(void \*)0,**

**(OS\_PRIO )APP\_CFG\_LED\_PRIO,**

**(CPU\_STK \*)&LEDStk[0],**

**(CPU\_STK\_SIZE )APP\_CFG\_TASK\_START\_STK\_SIZE\_LIMIT,**

**(CPU\_STK\_SIZE )APP\_CFG\_TASK\_START\_STK\_SIZE,**

**(OS\_MSG\_QTY )0u,**

**(OS\_TICK )0u,**

**(void \*)0,**

**(OS\_OPT )(OS\_OPT\_TASK\_STK\_CHK | OS\_OPT\_TASK\_STK\_CLR),**

**(OS\_ERR \*)&os\_err);**

**OSTaskCreate((OS\_TCB \*)&CountingTCB, /\* Create the start task \*/**

**(CPU\_CHAR \*)"Counting",**

**(OS\_TASK\_PTR ) Counting,**

**(void \*)0,**

**(OS\_PRIO )APP\_CFG\_COUNTING\_PRIO,**

**(CPU\_STK \*)&CountingStk[0],**

**(CPU\_STK\_SIZE )APP\_CFG\_TASK\_START\_STK\_SIZE\_LIMIT,**

**(CPU\_STK\_SIZE )APP\_CFG\_TASK\_START\_STK\_SIZE,**

**(OS\_MSG\_QTY )100u,**

**(OS\_TICK )0u,**

**(void \*)0,**

**(OS\_OPT )(OS\_OPT\_TASK\_STK\_CHK | OS\_OPT\_TASK\_STK\_CLR),**

**(OS\_ERR \*)&os\_err);**

**}**

**/\***

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

**\* App\_ObjCreate()**

**\***

**\* Description : Create application kernel objects tasks.**

**\***

**\* Argument(s) : none**

**\***

**\* Return(s) : none**

**\***

**\* Caller(s) : AppTaskStart()**

**\***

**\* Note(s) : none.**

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

**\*/**

**static void App\_ObjCreate (void)**

**{**

**}**

**void Polling(void)**

**{**

**while(DEF\_ON)**

**{**

**if(BTN1!=0)**

**{**

**OSTaskQPost(&CountingTCB, 4, 1u, OS\_OPT\_POST\_FIFO, &os\_err);**

**}**

**if(BTN2!=0)**

**{**

**OSTaskQPost(&CountingTCB, 3, 1u, OS\_OPT\_POST\_FIFO, &os\_err);**

**}**

**if(BTN3!=0)**

**{**

**OSTaskQPost(&CountingTCB, 1, 1u, OS\_OPT\_POST\_FIFO, &os\_err);**

**}**

**else**

**{**

**OSTaskSemPost(&LED\_TCB, OS\_OPT\_NONE, &os\_err );**

**}**

**if(SW1 != 0)**

**{**

**toggle\_value = 1;**

**}**

**else**

**{**

**toggle\_value = 0;**

**}**

**OSTimeDlyHMSM(0u,0u,0u,100u, OS\_OPT\_TIME\_HMSM\_STRICT, &os\_err);**

**Nop();**

**}**

**}**

**void LED(void)**

**{**

**int b;**

**CPU\_INT08U Temp;**

**while(DEF\_ON)**

**{**

**if(toggle\_value)**

**{**

**Temp = quantity;**

**}**

**else**

**{**

**Temp = sales;**

**}**

**for(b=1;b<=8;b++,Temp/=2)**

**{**

**LED\_Off(b);**

**if(Temp%2)**

**{**

**LED\_On(b);**

**}**

**}**

**OSTaskSemPend(0, OS\_OPT\_PEND\_BLOCKING, &ts, &os\_err);**

**}**

**}**

**/\*This task counts the amount of items that are sold. With each button**

**\*pressed dependant on the button. If BTN1 is pressed case 1 is used**

**\*which adds a “soda”.**

**void Counting(void)**

**{**

**CPU\_INT08U message;**

**while(DEF\_ON)**

**{**

**message = OSTaskQPend( 0, OS\_OPT\_PEND\_BLOCKING, &j, &ts, &os\_err);**

**switch (message)**

**{**

**case 4:**

**bottles++;**

**break;**

**case 3:**

**cans++;**

**break;**

**case 1:**

**sodas++;**

**break;**

**}**

**quantity++;**

**sales += message;**

**OSTaskSemPost(&LED\_TCB, OS\_OPT\_NONE, &os\_err);**

**}**

**}**

**/\* This is code that would have been used if interrupts did work out.**

**void BSP\_INT1Handler(void)**

**{**

**OS\_ERR err;**

**OSTaskQPost(&CountingTCB,3,1,OS\_OPT\_POST\_FIFO,&os\_err);**

**}**

**void BSP\_INT0Handler(void)**

**{**

**OS\_ERR err;**

**OSTaskQPost(&CountingTCB,4,1,OS\_OPT\_POST\_FIFO,&os\_err);**

**}**

**void BSP\_INT2Handler(void)**

**{**

**OS\_ERR err;**

**OSTaskQPost(&CountingTCB,1,1,OS\_OPT\_POST\_FIFO,&os\_err);**

**}\*/**