```
/*******************
Mark Moerdyk
First modification: 3/09/13
Last modification: 2/22/13
************************/

#include "includes.h"

#define DC1 (INT8U)0x11
#define DC2 (INT8U)0x12
#define DC3 (INT8U)0x13
#define DC4 (INT8U)0x14

/*********************************************************************
* Public Event Definitions
*********************************************************************/

/*********************************************************************
* Task Function Prototypes.
*   - Private if in the same module as startup task. Otherwise public.
*********************************************************************/
static void StartTask(void *p_arg);
static void UITask(void *p_arg);
static void TimeDispTask(void *p_arg);
static void TransmitTask(void *p_arg);
static void ReceiveTask(void *p_arg);
void sci_open(void);
void sci_open_int(void);
void sci_write(INT8U character);
INT8U sci_read(void);
/*********************************************************************
* Allocate task stack space.
*********************************************************************/
OS_STK  StartTaskStk[STARTTASK_STK_SIZE];
OS_STK  UITaskStk[UITASK_STK_SIZE];
OS_STK  TimeDispTaskStk[TIMEDISPTASK_STK_SIZE];
OS_STK  TransmitTaskStk[TRANSMITTASK_STK_SIZE];
OS_STK  ReceiveTaskStk[RECEIVETASK_STK_SIZE];
/*********************************************
*Global Variables
*********************************************/
OS_EVENT *TransmitFlag;
OS_EVENT *ReceiveFlag;
INT8U ReceivedString[25];
INT8U ReceiveVar;
INT8U AddNum1;
INT8U AddNum2;
INT8U AddNum3;
INT8U InTran = FALSE;
/*********************************************************************
* main()
Includes: Initialize OS, Key, and LCD
Creates start task
*********************************************************************/
void main(void)
{
    DBUG_PORT = 0x00;      //Initialize Debug bits
    DBUG_PORT_DIR = DB_OUTS;

    OSInit();

    TransmitFlag = OSSemCreate(0);
    ReceiveFlag = OSSemCreate(0);
    (void)OSTaskCreate(StartTask,         /* Create Startup Task        */
                (void *)0,
                (void *)&StartTaskStk[STARTTASK_STK_SIZE],
                STARTTASK_PRIO);

    OSStart();                      /* Start multitasking               */
}

/*********************************************************************
* STARTUP TASK - Prints out checksum and waits for c press. When C is pressed,
* starts LCD and Demo Task, then deletes itself
* Functions included: CalcChkSum, LcdDispStrg, DisplayCheckSum
* Creates: LCDDemoTask and DemoCntrlTask
*********************************************************************/
static void StartTask(void *p_arg)
{
    (void)p_arg;                            /* Avoid compiler warning     */
    OSTickInit();

    DBUG_PORT |= PP7;

    KeyInit();
    TimeInit();
    LcdInit(TRUE,TRUE,FALSE);
    SetTheTime();//for reset purposes
    (void)OSTaskCreate(UITask,         /* Create UITask   */
                (void *)0,
                (void *)&UITaskStk[UITASK_STK_SIZE],
                UITASK_PRIO);
    (void)OSTaskCreate(TimeDispTask,         /* Create TimeDispTask  */
                (void *)0,
                (void *)&TimeDispTaskStk[TIMEDISPTASK_STK_SIZE],
                TIMEDISPTASK_PRIO);
    (void)OSTaskCreate(TransmitTask,         /* Create UITask   */
                (void *)0,
                (void *)&TransmitTaskStk[TRANSMITTASK_STK_SIZE],
                TRANSMIT_PRIO);
    (void)OSTaskCreate(ReceiveTask,         /* Create UITask   */
                (void *)0,
                (void *)&ReceiveTaskStk[RECEIVETASK_STK_SIZE],
               RECEIVE_PRIO);

    DBUG_PORT &= ~PP7;
    (void)OSTaskDel(STARTTASK_PRIO);

    FOREVER()
    {
    }
}
/*************************************************************
*UITask - Task that waits for a keypress. If the # key is press, then jumps
to SetTheTime function. Else, waits for the # press*/
static void UITask(void *p_arg)
{
    INT8U keypress = 0;
    INT8U err;
    INT8U sendmessage;
    INT8U dbutton = TRUE;
    TIME displaytime;

    (void)p_arg;
    FOREVER()
    {
```

```
127:         DBUG_PORT &= ~PP6;
128:         keypress = KeyPend(0, &err);
129:         DBUG_PORT |= PP6;
130:
131:         if(keypress == '#')
132:         {
133:             DBUG_PORT |= PP6;
134:             SetTheTime();
135:             DBUG_PORT &= ~PP6;
136:         }
137:         else if (keypress == DC1)
138:         {
139:             DBUG_PORT |= PP6;
140:             InTran = TRUE; //if message received throw up symbol
141:             LcdShowLayer(CLOCK_LAYER);
142:             LcdHideLayer(DBUTTON_LAYER);
143:             LcdHideLayer(DISPLAY_LAYER);
144:             TypeText();
145:             InTran = FALSE;//out of transmit phase
146:             LcdShowLayer(DISPLAY_LAYER);
147:             TransmitCheck(&sendmessage);
148:             if(sendmessage == TRUE)
149:             {
150:                 OSSemPost(TransmitFlag);//Allows transmit to start
151:             }
152:             else
153:             {
154:             }
155:             sendmessage = FALSE;
156:             DBUG_PORT &= ~PP6;
157:         }
158:         else if (keypress == DC4)
159:         {
160:             DBUG_PORT |= PP6;
161:             if(dbutton == TRUE)
162:             {
163:
164:                 LcdShowLayer(DBUTTON_LAYER);
165:                 LcdHideLayer(CLOCK_LAYER);
166:                 LcdDispChar(1,1,DBUTTON_LAYER,AddNum1);
167:                 LcdDispChar(1,2,DBUTTON_LAYER,AddNum2);
168:                 LcdDispChar(1,3,DBUTTON_LAYER,AddNum3);
169:                 DispTimeStamp();//displays when message was received
170:                 dbutton = FALSE;
171:             }
172:             else
173:             {
174:                 dbutton = TRUE;
175:                 LcdHideLayer(DBUTTON_LAYER);
176:                 LcdShowLayer(CLOCK_LAYER);
177:             }
178:             DBUG_PORT &= ~PP6;
179:         }
180:         else
181:         {
182:         }
183:
184:     }
185:
186: }
187: /*************************************************
188: TimeDispTask - Takes the value of TimeOfClock, and displays it on the LCD
189: Functions: TimeGet, LCD
190: *************************************************/
191: static void TimeDispTask(void *p_arg)
192: {
193:     TIME displaytime;
194:     (void)p_arg;
195:     FOREVER()
196:     {
197:
198:         DBUG_PORT |= PP4;
199:         OSTimeDly(100);
200:         TimeGet(&displaytime);
201:         LcdDispTime(1,9,CLOCK_LAYER,displaytime.hr,displaytime.min,
202:                     displaytime.sec);
203:         DBUG_PORT &= ~PP4;
204:
205:     }
206: }
207: /*****************************************
208: TransmitTask - pends on the Transmit flag, when flag is posted
209: transmitts message typed.
210: uses: sci_write, sci_open, MessageCheckSum();
211: *****************************************/
212: static void TransmitTask(void *p_arg)
213: {
214:     INT8U err;
215:     INT8U var = 'M';
216:     INT8U sourcea = '1';
217:     INT8U sourceb = '1';
218:     INT8U sourcec = '7';
219:     INT8U send_message[16];
220:     INT8U counter = 0x00;
221:     INT8U checksum[2];
222:     (void)p_arg;
223:     sci_open();
224:
225:     FOREVER()
226:     {
227:         OSSemPend(TransmitFlag, 0, &err);
228:         DBUG_PORT |= PP1;
229:         sci_write(var);
230:         sci_write(sourcea);
231:         sci_write(sourceb);
232:         sci_write(sourcec);
233:         GetMessage(&send_message);
234:         while(counter != 0x10)
235:         {
236:             sci_write(send_message[counter]);
237:             counter++;
238:         }
239:         counter = 0x00;
240:         MessageCheckSum(&checksum,sourcea,sourceb,sourcec);
241:         sci_write(checksum[0]);
242:         sci_write(checksum[1]);
243:         DBUG_PORT &= ~PP1;
244:
245:     }
246:
247: }
248: /*****************************************
249:  ReceiveTask- pends on the interupt service routine, when condition
250:  is filled, fills up array, then checks if message is good. If good,
251:  post message, else CS ERROR
252:  Uses:sci_open_int,sci_read,ReceivedCheckSum
```

```
253: ******************************************/
254: static void ReceiveTask(void *p_arg)
255: {
256:     TIME get_time;
257:     INT8U input=0;
258:     INT8U err;
259:     INT8U transfer = 0x00;
260:     INT8U taken_message[16];
261:     INT8U sent_checksum[2];
262:     INT8U real_checksum[2];
263:     INT8U fill_array = FALSE;
264:
265:     (void)p_arg;
266:     sci_open_int();
267:
268:     FOREVER()
269:     {
270:         while(input != 22)
271:         {
272:             OSSemPend(ReceiveFlag, 0, &err);
273:             if(ReceiveVar == 'M')//only m
274:             {
275:                 fill_array = TRUE;
276:             }
277:             else{}
278:             if(fill_array == TRUE)
279:             {
280:                 ReceivedString[input] = ReceiveVar;
281:                 input++;
282:             }
283:             else{}
284:
285:         }
286:         DBUG_PORT |= PP0;
287:         fill_array = FALSE;
288:         LcdDispClear(DISPLAY_LAYER);
289:         if (InTran == TRUE)
290:         {
291:             LcdDispChar(1,1,MESSAGE_LAYER,'!');
292:         }
293:         else{}
294:         input = 0x04;
295:         AddNum1 = ReceivedString[1];
296:         AddNum2 = ReceivedString[2];
297:         AddNum3 = ReceivedString[3];
298:
299:         while(transfer != 16)
300:         {
301:             taken_message[transfer] = ReceivedString[input];
302:             transfer++;
303:             input++;
304:         }
305:         transfer = 0x00;
306:         sent_checksum[0] = ReceivedString[20];
307:         sent_checksum[1] = ReceivedString[21];
308:         ReceivedCheckSum(&real_checksum,&taken_message,AddNum1,AddNum2,
309:                          AddNum3);
310:         if((real_checksum[1] == sent_checksum[1]) &&
311:           (real_checksum[0] == sent_checksum[0]))
312:         {
313:             while(transfer != 16)
314:             {
315:                 if(taken_message[transfer] != 0)
316:                 {
317:                     LcdDispChar(2,(transfer + 1),DISPLAY_LAYER,
318:                                 taken_message[transfer]);
319:
320:                 }
321:                 else
322:                 {
323:                     LcdDispChar(2,(transfer + 1),DISPLAY_LAYER,' ');
324:                 }
325:                 transfer++;
326:             }
327:         }
328:         else
329:         {
330:             LcdDispChar(2,1,DISPLAY_LAYER,'C');
331:             LcdDispChar(2,2,DISPLAY_LAYER,'S');
332:             LcdDispChar(2,3,DISPLAY_LAYER,' ');
333:             LcdDispChar(2,4,DISPLAY_LAYER,'E');
334:             LcdDispChar(2,5,DISPLAY_LAYER,'r');
335:             LcdDispChar(2,6,DISPLAY_LAYER,'r');
336:             LcdDispChar(2,7,DISPLAY_LAYER,'o');
337:             LcdDispChar(2,8,DISPLAY_LAYER,'r');
338:         }
339:         transfer = 0x00;
340:         input = 0x00;
341:         GetReceiveTime();
342:         DBUG_PORT &= ~PP0;
343:     }
344: }
345: //interupt waits for a read from the port
346: ISR OCOIsr(void)
347: {
348:     OS_ISR_ENTER();
349:     ReceiveVar = sci_read();
350:     OSSemPost(ReceiveFlag);
351:     OSIntExit();
352: }
```