```
 1: /*******************
 2: Mark Moerdyk
 3: First modification: 2/18/13
 4: Last modification: 2/22/13
 5: ************************/
 6:
 7: #include "includes.h"
 8:
 9: #define DC1 (INT8U)0x11
10: #define DC2 (INT8U)0x12
11: #define DC3 (INT8U)0x13
12: #define DC4 (INT8U)0x14
13:
14: OS_EVENT *SecFlag;
15: /*********************************************************************
16: * Task Function Prototypes.
17: *   - Private if in the same module as startup task. Otherwise public.
18: *********************************************************************/
19: static void ClockTask(void *p_arg);
20: void TimeGet(TIME *ltime);
21: void TimeSet(TIME *ltime);
22: void TimeInit(void);
23: void ClockTimerFnct(void *ptmr, void *callback_arg);
24:
25: /*********************************************************************
26: * Allocate task stack space.
27: *********************************************************************/
28: OS_EVENT *ClockMutexKey;
29: OS_TMR *ClockTimer;
30: OS_STK  ClockTaskStk[CLOCKTASK_STK_SIZE];
31: /***********************************************
32: *Global Variables
33: ***********************************************/
34:
35: static TIME TimeOfDay;
36:
37: /***********************************************
38: *Clocktask - counts the number of clock cycles like a 12 hour clock
39: * uses a secflag in order to count a 1 second cycle
40: Initialize: TimeOfDay
41: ***********************************************/
42: static void ClockTask(void *p_arg)
43: {
44:     INT8U error;
45:     INT8U key;
46:     INT8U keypress;
47:     INT8U err;
48:     (void)p_arg;
49:     FOREVER()
50:     {
51:         DBUG_PORT &= ~PP3;
52:         OSSemPend(SecFlag, 0 ,&err);
53:         DBUG_PORT |= PP3;
54:         TimeOfDay.sec = TimeOfDay.sec + 1;
55:
56:         if(TimeOfDay.sec > 0x3B && TimeOfDay.min < 0x3B && TimeOfDay.hr < 0x0D)
57:         {
58:             TimeOfDay.sec = 0x00;
59:             TimeOfDay.min++;
60:         }
61:         else if(TimeOfDay.sec > 0x3B && TimeOfDay.min > 0x3B && TimeOfDay.hr < 0x0D)
62:         {
63:             TimeOfDay.sec = 0x00;
64:             TimeOfDay.min = 0x00;
65:             TimeOfDay.hr++;
66:         }
67:         else if (TimeOfDay.sec > 0x3B && TimeOfDay.min == 0x3B
68:                  && TimeOfDay.hr == 0x0C)
69:         {
70:             TimeOfDay.sec = 0x00;
71:             TimeOfDay.min = 0x00;
72:             TimeOfDay.hr = 0x01;
73:         }
74:         else
75:         {
76:         }
77:     }
78: }
79: /*******************************************************
80: TimeInit - function that initializes timer, mutex, and clock task.
81: * sets values to timeOfDay if reset button is hit
82: Creates: ClockTimer, ClockMutexKey, SecFlag
83: *******************************************************8*/
84: void TimeInit(void)
85: {
86:
87:
88:     INT8U err;
89:     TimeOfDay.hr = 0x0C;
90:     TimeOfDay.min = 0x00;
91:     TimeOfDay.sec = 0x00;
92:     LcdDispTime(TimeOfDay.hr,TimeOfDay.min,TimeOfDay.sec);
93:
94:     ClockTimer = OSTmrCreate(0,
95:                              10,
96:                              OS_TMR_OPT_PERIODIC,
97:                              ClockTimerFnct,
98:                              (void *)0,
99:                              "Clock Timer ",
100:                             &err);
101:    OSTmrStart(ClockTimer, &err);
102:
103:    ClockMutexKey = (CLOCK_PIP, &err);
104:
105:
106:    (void)OSTaskCreate(ClockTask,
107:                (void *)0,
108:                (void *)&ClockTaskStk[CLOCKTASK_STK_SIZE],
109:                CLOCKTASK_PRIO);
110:
111:    SecFlag = OSSemCreate(0);
112: }
113:  /***************************************************
114:  TimeSet - takes the programmed time of the clock, and
115:  * sets it to the TimeofDay
116:  Passes in: ltime
117:  Passes out: nothing
118:  ***************************************************/
119: void TimeSet(TIME *ltime)
120: {
121:     INT8U err;
122:     OSMutexPend(ClockMutexKey,0,&err);
123:     TimeOfDay = *ltime;
124:     OSMutexPost(ClockMutexKey);
125: }
126: /***************************************************
```

```
127: TimeGet- sets the value of TimeofDay to ltime to be displyed
128: on the LCD
129: Passes in: nothing
130: Passes out : TimeOfDay
131: ************************************************/
132: void TimeGet(TIME *ltime)
133: {
134:     INT8U err;
135:     OSMutexPend(ClockMutexKey,0,&err);
136:     *ltime = TimeOfDay ;
137:     OSMutexPost(ClockMutexKey);
138: }
139: /************************************************
140: * ClockTimerFnct - Gets called from the OSTmrCreate to
141: * post the SecFlag created in the TimeInit function
142: ************************************************/
143: void ClockTimerFnct(void *ptmr, void *callback_arg)
144: {
145:     OSSemPost(SecFlag);
146: }
```