

```

1: /*****
2: Mark Moerdyk
3: First modification: 2/14/13
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: /*****
15: * Public Event Definitions
16: *****/
17:
18: /*****
19: * Task Function Prototypes.
20: * - Private if in the same module as startup task. Otherwise public.
21: *****/
22: static void StartTask(void *p_arg);
23: static void UITask(void *p_arg);
24: static void TimeDispTask(void *p_arg);
25: void SetTheTime(void);
26:
27: /*****
28: * Allocate task stack space.
29: *****/
30: OS_STK StartTaskStk[STARTTASK_STK_SIZE];
31: OS_STK UITaskStk[UITASK_STK_SIZE];
32: OS_STK TimeDispTaskStk[TIMEDISPTASK_STK_SIZE];
33: /*****
34: *Global Variables
35: *****/
36:
37: typedef enum {INITIALSETUP, TENHRPLACE, ONEHRPLACE, TENMINPLACE, ONEMINPLACE,
38:               TENSECPLACE, ONESECPLACE, VALUEGOESTHROUGH, BACKTOORIGINAL,
39:               }CLKSTATES;
40: /*****
41: * main()
42: Includes: Initialize OS, Key, and LCD
43: Creates start task
44: *****/
45: void main(void)
46: {
47:     DEBUG_PORT = 0x00; //Initialize Debug bits
48:     DEBUG_PORT_DIR = DB_OUTS;
49:
50:     OSInit(); // Initialize uC/OS-II */
51:     KeyInit();
52:     LcdInit();
53:
54:     (void)OSTaskCreate(StartTask, /* Create Startup Task */
55:                       (void *)0,
56:                       (void *)&StartTaskStk[STARTTASK_STK_SIZE],
57:                       STARTTASK_PRIO);
58:
59:     OSStart(); // Start multitasking */
60: }
61:
62: /*****
63: * STARTUP TASK - Prints out checksum and waits for c press. When C is pressed,

```

```

64: * starts LCD and Demo Task, then deletes itself
65: * Functions included: CalcChkSum, LcdDispStrg, DisplayChecksum
66: * Creates: LCDDemoTask and DemoCntrlTask
67: *****/
68: static void StartTask(void *p_arg)
69: {
70:     (void)p_arg; // Avoid compiler warning */
71:     OSTickInit();
72:     LcdClrDisp();
73:     LcdMoveCursor(1,5);
74:
75:     DEBUG_PORT |= PP7;
76:
77:     TimeInit();
78:     (void)OSTaskCreate(UITask, /* Create UITask */
79:                       (void *)0,
80:                       (void *)&UITaskStk[UITASK_STK_SIZE],
81:                       UITASK_PRIO);
82:     (void)OSTaskCreate(TimeDispTask, /* Create TimeDispTask */
83:                       (void *)0,
84:                       (void *)&TimeDispTaskStk[TIMEDISPTASK_STK_SIZE],
85:                       TIMEDISPTASK_PRIO);
86:
87:     DEBUG_PORT &= ~PP7;
88:     (void)OSTaskDel(STARTTASK_PRIO);
89:     FOREVER()
90:     {
91:         //do nothing
92:     }
93: }
94: /*****
95: *UITask - Task that waits for a keypress. If the # key is press, then jumps to
96: SetTheTime function. Else, waits for the # press*/
97: static void UITask(void *p_arg)
98: {
99:     INT8U keypress = 0;
100:    INT8U key;
101:    INT8U err;
102:
103:    (void)p_arg;
104:    FOREVER()
105:    {
106:        DEBUG_PORT &= ~PP6;
107:        keypress = KeyPend(key, &err);
108:        DEBUG_PORT |= PP6;
109:        if(keypress == '#')
110:        {
111:            DEBUG_PORT |= PP6;
112:            SetTheTime();
113:            DEBUG_PORT &= ~PP6;
114:        }
115:        else
116:        {
117:        }
118:    }
119: }
120: }
121: /*****
122: TimeDispTask - Takes the value of TimeOfClock, and displays it on the LCD
123: Functions: TimeGet, LCD
124: *****/
125: static void TimeDispTask(void *p_arg)
126: {

```

## main.c

```

127:     TIME displaytime;
128:     (void)p_arg;
129:
130:     FOREVER()
131:     {
132:         DEBUG_PORT |= PP4;
133:         TimeGet(&displaytime);
134:         LcdMoveCursor(1,5);
135:         LcdDispTime(displaytime.hr,displaytime.min,displaytime.sec);
136:         DEBUG_PORT &= ~PP4;
137:     }
138: }
139: }
140: /*****
141: SetTheTime - Goes through each of the six different places that can be
142: programmed for time, and when done, sets the time of the programmed value
143: equal to the TimeOfDay time. If C is pressed, nothing happens.
144: Functions: TimeSet(), OSTaskSuspend(), OSTaskResume()
145: *****/
146: void SetTheTime(void)
147: {
148:     TIME changetime;
149:     CLKSTATES curstate = INITIALSETUP;
150:     INT8U err;
151:     INT8U keypress = 0;
152:     INT8U key;
153:     INT8U hrtenval = 0x00;
154:     INT8U hroneval = 0x00;
155:     INT8U mintenval = 0x00;
156:     INT8U minoneval = 0x00;
157:     INT8U sectenval = 0x00;
158:     INT8U seconeval = 0x00;
159:     INT8U finishset = 0x00;
160:     INT8U tenhrset = FALSE;
161:     INT8U onehrset = FALSE;
162:     INT8U tenminset = FALSE;
163:     INT8U oneminset = FALSE;
164:     INT8U tensecset = FALSE;
165:     INT8U oneseccset = FALSE;
166:     INT8U remainder;
167:     INT8U hourset = FALSE;
168:     INT8U onepressed = FALSE;
169:     INT8U zeropressed = FALSE;
170:
171:     OSTaskSuspend(TIMEDISPTASK_PRIO);
172:     TimeGet(&changetime);
173:     LcdMoveCursor(1,5);
174:
175:     keypress = KeyPend(key, &err);
176:
177:     while((finishset != 0x01) && (finishset != 0x02))
178:     {
179:
180:         switch(curstate)
181:         {
182:             case(INITIALSETUP):
183:
184:                 LcdMoveCursor(1,5);
185:                 LcdCursor(TRUE,TRUE);
186:                 curstate = TENHRPLACE;
187:                 break;
188:
189:             case(TENHRPLACE):

```

```

190:
191:         keypress = KeyPend(0, &err);
192:         while((keypress != '1') && (keypress != '0')&& (keypress != DC3)
193:             && (keypress != DC1))
194:         {
195:             keypress = KeyPend(0, &err);
196:         } //do nothing
197:         if(keypress == '1')
198:         {
199:             if ((changetime.hr <= 0x09)&& (changetime.hr > 0x02)
200:                 && (hourset == FALSE))
201:             {
202:                 curstate = TENHRPLACE;
203:             }
204:             else if(hroneval <= 0x02)
205:             {
206:                 LcdDispChar('1');
207:                 hrtenval = 0x0A;
208:                 tenhrset = TRUE;
209:                 hourset = TRUE;
210:                 onepressed = TRUE;
211:                 curstate = ONEHRPLACE;
212:             }
213:             else
214:             {
215:                 curstate = TENHRPLACE;
216:             }
217:         }
218:         else if(keypress == DC1)
219:         {
220:             curstate = VALUEGOESTHROUGH;
221:         }
222:         else if(keypress == DC3)
223:         {
224:             curstate = BACKTOORIGINAL;
225:         }
226:
227:         else
228:         {
229:             if ((hroneval == 0x00) && (onehrset == TRUE))
230:             {
231:                 curstate = TENHRPLACE;
232:             }
233:             else
234:             {
235:                 LcdDispChar('0');
236:                 hrtenval = 0x00;
237:                 tenhrset = TRUE;
238:                 hourset = TRUE;
239:                 zeropressed = TRUE;
240:                 curstate = ONEHRPLACE;
241:             }
242:         }
243:         break;
244:
245:     case(ONEHRPLACE):
246:
247:         LcdMoveCursor(1,6);
248:         keypress = KeyPend(0, &err);
249:         while((keypress != '0') && (keypress != '1') && (keypress != '2')
250:             && (keypress != '3') && (keypress != '4') && (keypress != '5')
251:             && (keypress != '6') && (keypress != '7') && (keypress != '8')
252:             && (keypress != '9') && (keypress != DC2) && (keypress != DC3)

```

## main.c

```
253:         && (keypress != DC1))
254:     {
255:         keypress = KeyPend(0, &err);
256:     }
257: if(keypress == '0')
258: {
259:     if (hrtenval == 0x00)
260:     {
261:         curstate = ONEHRPLACE;
262:     }
263:     else
264:     {
265:         LcdDispChar('0');
266:         hroneval = 0x00;
267:         onehrset = TRUE;
268:         curstate = TENMINPLACE;
269:     }
270: }
271: else if(keypress == '1')
272: {
273:     LcdDispChar('1');
274:     hroneval = 0x01;
275:     onehrset = TRUE;
276:     curstate = TENMINPLACE;
277: }
278: else if(keypress == '2')
279: {
280:     LcdDispChar('2');
281:     hroneval = 0x02;
282:     onehrset = TRUE;
283:     curstate = TENMINPLACE;
284: }
285: else if (keypress == '3')
286: {
287:     if (hrtenval == 0x0A)
288:     {
289:         curstate = ONEHRPLACE;
290:     }
291:     else
292:     {
293:         LcdDispChar('3');
294:         hroneval = 0x03;
295:         onehrset = TRUE;
296:         curstate = TENMINPLACE;
297:     }
298: }
299: }
300: else if (keypress == '4')
301: {
302:     if (hrtenval == 0x0A)
303:     {
304:         curstate = ONEHRPLACE;
305:     }
306:     else
307:     {
308:         LcdDispChar('4');
309:         hroneval = 0x04;
310:         onehrset = TRUE;
311:         curstate = TENMINPLACE;
312:     }
313: }
314: }
315: else if (keypress == '5')
```

```
316: {
317:     if (hrtenval == 0x0A)
318:     {
319:         curstate = ONEHRPLACE;
320:     }
321:     else
322:     {
323:         LcdDispChar('5');
324:         hroneval = 0x05;
325:         onehrset = TRUE;
326:         curstate = TENMINPLACE;
327:     }
328: }
329: }
330: else if (keypress == '6')
331: {
332:     if (hrtenval == 0x0A)
333:     {
334:         curstate = ONEHRPLACE;
335:     }
336:     else
337:     {
338:         LcdDispChar('6');
339:         hroneval = 0x06;
340:         onehrset = TRUE;
341:         curstate = TENMINPLACE;
342:     }
343: }
344: }
345: else if (keypress == '7')
346: {
347:     if (hrtenval == 0x0A)
348:     {
349:         curstate = ONEHRPLACE;
350:     }
351:     else
352:     {
353:         LcdDispChar('7');
354:         hroneval = 0x07;
355:         onehrset = TRUE;
356:         curstate = TENMINPLACE;
357:     }
358: }
359: }
360: else if (keypress == '8')
361: {
362:     if (hrtenval == 0x0A)
363:     {
364:         curstate = ONEHRPLACE;
365:     }
366:     else
367:     {
368:         LcdDispChar('8');
369:         hroneval = 0x08;
370:         onehrset = TRUE;
371:         curstate = TENMINPLACE;
372:     }
373: }
374: }
375: else if (keypress == '9')
376: {
377:     if (hrtenval == 0x0A)
378:     {
```

## main.c

```
379:         curstate = ONEHRPLACE;
380:     }
381:     else
382:     {
383:         LcdDispChar('9');
384:         hroneval = 0x09;
385:         onehrset = TRUE;
386:         curstate = TENMINPLACE;
387:     }
388:
389: }
390:
391: else if(keypress == DC1)
392: {
393:     curstate = VALUEGOESTHROUGH;
394: }
395: else if(keypress == DC3)
396: {
397:     curstate = BACKTOORIGINAL;
398: }
399: else
400: {
401:     LcdBSpace();
402:     curstate = TENHRPLACE;
403: }
404: OTimeDly(100);
405: break;
406:
407: case(TENMINPLACE):
408:
409:     LcdMoveCursor(1,8);
410:     keypress = KeyPend(0, &err);
411:     while((keypress != '0') && (keypress != '1') && (keypress != '2')
412:           && (keypress != '3') && (keypress != '4') && (keypress != '5')
413:           && (keypress != DC2) && (keypress != DC3)
414:           && (keypress != DC1))
415:     {
416:         keypress = KeyPend(0, &err);
417:     }
418:     if(keypress == '0')
419:     {
420:         LcdDispChar('0');
421:         mininterval = 0x00;
422:         tenminset = TRUE;
423:         curstate = ONEMINPLACE;
424:     }
425:     else if(keypress == '1')
426:     {
427:         LcdDispChar('1');
428:         mininterval = 0x0A;
429:         tenminset = TRUE;
430:         curstate = ONEMINPLACE;
431:     }
432:     else if(keypress == '2')
433:     {
434:         LcdDispChar('2');
435:         mininterval = 0x14;
436:         tenminset = TRUE;
437:         curstate = ONEMINPLACE;
438:     }
439:     else if(keypress == '3')
440:     {
441:         LcdDispChar('3');
```

```
442:         mininterval = 0x1E;
443:         tenminset = TRUE;
444:         curstate = ONEMINPLACE;
445:     }
446:     else if(keypress == '4')
447:     {
448:         LcdDispChar('4');
449:         mininterval = 0x28;
450:         tenminset = TRUE;
451:         curstate = ONEMINPLACE;
452:     }
453:     else if(keypress == '5')
454:     {
455:         LcdDispChar('5');
456:         mininterval = 0x32;
457:         tenminset = TRUE;
458:         curstate = ONEMINPLACE;
459:     }
460:     else if(keypress == DC1)
461:     {
462:         curstate = VALUEGOESTHROUGH;
463:     }
464:     else if(keypress == DC3)
465:     {
466:         curstate = BACKTOORIGINAL;
467:     }
468:     else
469:     {
470:         curstate = ONEHRPLACE;
471:     }
472:     break;
473:
474: case(ONEMINPLACE):
475:
476:     LcdMoveCursor(1,9);
477:     keypress = KeyPend(0, &err);
478:     while((keypress != '0') && (keypress != '1') && (keypress != '2')
479:           && (keypress != '3') && (keypress != '4') && (keypress != '5')
480:           && (keypress != '6') && (keypress != '7') && (keypress != '8')
481:           && (keypress != '9') && (keypress != DC2) && (keypress != DC3)
482:           && (keypress != DC1))
483:     {
484:         keypress = KeyPend(0, &err);
485:     }
486:     if(keypress == '0')
487:     {
488:         LcdDispChar('0');
489:         minoneval = 0x00;
490:         oneminset = TRUE;
491:         curstate = TENSECPLACE;
492:     }
493:     else if(keypress == '1')
494:     {
495:         LcdDispChar('1');
496:         minoneval = 0x01;
497:         oneminset = TRUE;
498:         curstate = TENSECPLACE;
499:     }
500:     else if(keypress == '2')
501:     {
502:         LcdDispChar('2');
503:         minoneval = 0x02;
504:         oneminset = TRUE;
```

```
505:         curstate = TENSECPLACE;
506:     }
507:     else if(keypress == '3')
508:     {
509:         LcdDispChar('3');
510:         minoneval = 0x03;
511:         oneminset = TRUE;
512:         curstate = TENSECPLACE;
513:     }
514:     else if(keypress == '4')
515:     {
516:         LcdDispChar('4');
517:         minoneval = 0x04;
518:         oneminset = TRUE;
519:         curstate = TENSECPLACE;
520:     }
521:     else if(keypress == '5')
522:     {
523:         LcdDispChar('5');
524:         minoneval = 0x05;
525:         oneminset = TRUE;
526:         curstate = TENSECPLACE;
527:     }
528:     else if(keypress == '6')
529:     {
530:         LcdDispChar('6');
531:         minoneval = 0x06;
532:         oneminset = TRUE;
533:         curstate = TENSECPLACE;
534:     }
535:     else if(keypress == '7')
536:     {
537:         LcdDispChar('7');
538:         minoneval = 0x07;
539:         oneminset = TRUE;
540:         curstate = TENSECPLACE;
541:     }
542:     else if(keypress == '8')
543:     {
544:         LcdDispChar('8');
545:         minoneval = 0x08;
546:         oneminset = TRUE;
547:         curstate = TENSECPLACE;
548:     }
549:     else if(keypress == '9')
550:     {
551:         LcdDispChar('9');
552:         minoneval = 0x09;
553:         oneminset = TRUE;
554:         curstate = TENSECPLACE;
555:     }
556:     else if(keypress == DC1)
557:     {
558:         curstate = VALUEGOESTHROUGH;
559:     }
560:     else if(keypress == DC3)
561:     {
562:         curstate = BACKTOORIGINAL;
563:     }
564:     else
565:     {
566:         curstate = TENMINPLACE;
567:     }
```

```
568:         break;
569:
570:     case(TENSECPLACE):
571:
572:         LcdMoveCursor(1,11);
573:         keypress = KeyPend(0, &err);
574:         while((keypress != '0') && (keypress != '1') && (keypress != '2')
575:             && (keypress != '3') && (keypress != '4') && (keypress != '5')
576:             && (keypress != DC2) && (keypress != DC3)
577:             && (keypress != DC1))
578:         {
579:             keypress = KeyPend(0, &err);
580:         }
581:         if(keypress == '0')
582:         {
583:             LcdDispChar('0');
584:             sectenval = 0x00;
585:             tensecset = TRUE;
586:             curstate = ONESECPLACE;
587:         }
588:         else if(keypress == '1')
589:         {
590:             LcdDispChar('1');
591:             sectenval = 0x0A;
592:             tensecset = TRUE;
593:             curstate = ONESECPLACE;
594:         }
595:         else if(keypress == '2')
596:         {
597:             LcdDispChar('2');
598:             sectenval = 0x14;
599:             tensecset = TRUE;
600:             curstate = ONESECPLACE;
601:         }
602:         else if(keypress == '3')
603:         {
604:             LcdDispChar('3');
605:             sectenval = 0x1E;
606:             tensecset = TRUE;
607:             curstate = ONESECPLACE;
608:         }
609:         else if(keypress == '4')
610:         {
611:             LcdDispChar('4');
612:             sectenval = 0x28;
613:             tensecset = TRUE;
614:             curstate = ONESECPLACE;
615:         }
616:         else if(keypress == '5')
617:         {
618:             LcdDispChar('5');
619:             sectenval = 0x32;
620:             tensecset = TRUE;
621:             curstate = ONESECPLACE;
622:         }
623:         else if(keypress == DC1)
624:         {
625:             curstate = VALUEGOESTHROUGH;
626:         }
627:         else if(keypress == DC3)
628:         {
629:             curstate = BACKTOORIGINAL;
630:         }
```

## main.c

```

631:         else
632:         {
633:             curstate = ONEMINPLACE;
634:         }
635:         break;
636:
637:     case(ONESECPLACE):
638:
639:         LcdMoveCursor(1,12);
640:         keypress = KeyPend(0, &err);
641:         while((keypress != '0') && (keypress != '1') && (keypress != '2')
642:             && (keypress != '3') && (keypress != '4') && (keypress != '5')
643:             && (keypress != '6') && (keypress != '7') && (keypress != '8')
644:             && (keypress != '9') && (keypress != DC2) && (keypress != DC3)
645:             && (keypress != DC1))
646:         {
647:             keypress = KeyPend(0, &err);
648:         }
649:
650:         if(keypress == '0')
651:         {
652:             LcdDispChar('0');
653:             seconeval = 0x00;
654:             oneseccset = TRUE;
655:             curstate = VALUEGOESTHROUGH;
656:         }
657:         else if(keypress == '1')
658:         {
659:             LcdDispChar('1');
660:             seconeval = 0x01;
661:             oneseccset = TRUE;
662:             curstate = VALUEGOESTHROUGH;
663:         }
664:         else if(keypress == '2')
665:         {
666:             LcdDispChar('2');
667:             seconeval = 0x02;
668:             oneseccset = TRUE;
669:             curstate = VALUEGOESTHROUGH;
670:         }
671:         else if(keypress == '3')
672:         {
673:             LcdDispChar('3');
674:             oneseccset = TRUE;
675:             seconeval = 0x03;
676:             curstate = VALUEGOESTHROUGH;
677:         }
678:         else if(keypress == '4')
679:         {
680:             LcdDispChar('4');
681:             seconeval = 0x04;
682:             oneseccset = TRUE;
683:             curstate = VALUEGOESTHROUGH;
684:         }
685:         else if(keypress == '5')
686:         {
687:             LcdDispChar('5');
688:             seconeval = 0x05;
689:             oneseccset = TRUE;
690:             curstate = VALUEGOESTHROUGH;
691:         }
692:         else if(keypress == '6')
693:         {

```

```

694:             LcdDispChar('6');
695:             seconeval = 0x06;
696:             oneseccset = TRUE;
697:             curstate = VALUEGOESTHROUGH;
698:         }
699:         else if(keypress == '7')
700:         {
701:             LcdDispChar('7');
702:             seconeval = 0x07;
703:             oneseccset = TRUE;
704:             curstate = VALUEGOESTHROUGH;
705:         }
706:         else if(keypress == '8')
707:         {
708:             LcdDispChar('8');
709:             seconeval = 0x08;
710:             oneseccset = TRUE;
711:             curstate = VALUEGOESTHROUGH;
712:         }
713:         else if(keypress == '9')
714:         {
715:             LcdDispChar('9');
716:             seconeval = 0x09;
717:             oneseccset = TRUE;
718:             curstate = VALUEGOESTHROUGH;
719:         }
720:         else if(keypress == DC1)
721:         {
722:             curstate = VALUEGOESTHROUGH;
723:         }
724:         else if(keypress == DC3)
725:         {
726:             curstate = BACKTOORIGINAL;
727:         }
728:         else
729:         {
730:             curstate = TENSECPLACE;
731:         }
732:         break;
733:
734:     case(VALUEGOESTHROUGH):
735:
736:         finishset = 0x01;
737:         break;
738:
739:     case(BACKTOORIGINAL):
740:         finishset = 0x02;
741:         break;
742:
743:     default:
744:         break;
745:     }
746:
747: }
748: //sends the value and puts it in TimeSet()
749: if(finishset == 0x01)
750: {
751:     if(onesecset == TRUE)
752:     {
753:         changetime.hr = hrtenval + hroneval;
754:         changetime.min = mintenval + minoneval;
755:         changetime.sec = sectenval + seconeval;
756:     }

```

```
757:     else if((tensecset == TRUE) && (oneseccset == FALSE))
758:     {
759:         changetime.hr = hrtenval + hroneval;
760:         changetime.min = mintenval + minoneval;
761:         if( changetime.sec >= 0x32)
762:         {
763:             remainder = changetime.sec - 0x32;
764:         }
765:         else if(changetime.sec >= 0x28)
766:         {
767:             remainder = changetime.sec - 0x28;
768:         }
769:         else if(changetime.sec >= 0x1E)
770:         {
771:             remainder = changetime.sec - 0x1E;
772:         }
773:         else if(changetime.sec >=0x14)
774:         {
775:             remainder = changetime.sec - 0x14;
776:         }
777:         else if(changetime.sec >= 0x0A)
778:         {
779:             remainder = changetime.sec - 0x0A;
780:         }
781:         else
782:         {
783:             remainder = changetime.sec;
784:         }
785:         changetime.sec = sectenval + remainder;
786:     }
787:     else if((oneminset == TRUE) && (tensecset == FALSE))
788:     {
789:         changetime.hr = hrtenval + hroneval;
790:         changetime.min = mintenval + minoneval;
791:     }
792:     else if((tenminset == TRUE) && (oneminset == FALSE))
793:     {
794:         changetime.hr = hrtenval + hroneval;
795:         if( changetime.min >= 0x32)
796:         {
797:             remainder = changetime.min - 0x32;
798:         }
799:         else if(changetime.min >= 0x28)
800:         {
801:             remainder = changetime.min - 0x28;
802:         }
803:         else if(changetime.min >= 0x1E)
804:         {
805:             remainder = changetime.min - 0x1E;
806:         }
807:         else if(changetime.min >=0x14)
808:         {
809:             remainder = changetime.min - 0x14;
810:         }
811:         else if(changetime.min >= 0x0A)
812:         {
813:             remainder = changetime.min - 0x0A;
814:         }
815:         else
816:         {
817:             remainder = changetime.min;
818:         }
819:         changetime.min = mintenval+ remainder;
```

```
820:
821:     }
822:     else if((onehrset == TRUE) && (tenminset == FALSE))
823:     {
824:         changetime.hr = hrtenval + hroneval;
825:     }
826:     else if((tenhrset == TRUE) && (onehrset == FALSE) && (zeropressed == TRUE))
827:     {
828:         remainder = changetime.hr - 0x0A;
829:         changetime.hr = remainder + hrtenval;
830:     }
831:     else if((tenhrset == TRUE) && (onehrset == FALSE) && (onepressed == TRUE))
832:     {
833:         changetime.hr = changetime.hr + hrtenval;
834:     }
835:     else
836:     {
837:     }
838:     TimeSet(&changetime);
839: }
840: else
841: {
842: } //nothing
843: LcdMoveCursor(1,5);
844: LcdCursor(FALSE,FALSE);
845: OSTaskResume(TIMEDISPTASK_PRIO);
846: }
```