

```

1  /*****
2  /*
3  /*
4  /*
5  /*
6  /*
7  /*
8  /*****
9
10 /*
11 This file contains the functions for processing menu entries for the
12 Digital Oscilloscope project. These functions take care of maintaining the
13 menus and handling menu updates for the system. The functions included
14 are:
15     clear_menu      - remove the menu from the display
16     display_menu    - display the menu
17     init_menu       - initialize menus
18     menu_entry_left - take care of <Left> key for a menu entry
19     menu_entry_right - take care of <Right> key for a menu entry
20     next_entry      - next menu entry
21     previous_entry  - previous menu entry
22     refresh_menu    - re-display the menu if currently being displayed
23     reset_menu      - reset the current selection to the top of the menu
24
25 The local functions included are:
26     display_entry    - display a menu entry (including option setting)
27
28 The locally global variable definitions included are:
29     menu             - the menu
30     menu_display     - whether or not the menu is currently displayed
31     menu_entry       - the currently selected menu entry
32
33
34 Revision History
35     3/8/94   Glen George   Initial revision.
36     3/9/94   Glen George   Changed position of const keyword in array
37                     declarations involving pointers.
38     3/13/94  Glen George   Updated comments.
39     3/13/94  Glen George   Added display_entry function to output a menu
40                     entry and option setting to the LCD (affects
41                     many functions).
42     3/13/94  Glen George   Changed calls to set_status due to changing
43                     enum scale_status definition.
44     3/13/94  Glen George   No longer clear the menu area before
45                     restoring the trace in clear_menu() (not
46                     needed).
47     3/17/97  Glen George   Updated comments.
48     3/17/97  Glen George   Fixed minor bug in reset_menu().
49     3/17/97  Glen George   When initializing the menu in init_menu(),
50                     set the delay to MIN_DELAY instead of 0 and
51                     trigger to a middle value instead of
52                     MIN_TRG_LEVEL_SET.
53     5/3/06   Glen George   Changed to a more appropriate constant in
54                     display_entry().
55     5/3/06   Glen George   Updated comments.
56     5/9/06   Glen George   Changed menus to handle a list for mode and
57                     scale (move up and down list), instead of
58                     toggling values.
59 */
60
61
62
63 /* library include files */
64 /* none */
65
66 /* local include files */
67 #include "scopedef.h"
68 #include "lcdout.h"
69 #include "menu.h"
70 #include "menuact.h"
71 #include "tracutil.h"
72
73
74
75

```

```

76 /* local function declarations */
77 static void display_entry(int, int);      /* display a menu entry and its setting */
78
79
80
81
82 /* locally global variables */
83 static int menu_display;      /* TRUE if menu is currently displayed */
84
85 const static struct menu_item menu[] =      /* the menu */
86 { { "Mode", 0, 4, display_mode },
87   { "Scale", 0, 5, display_scale },
88   { "Sweep", 0, 5, display_sweep },
89   { "Trigger", 0, 7, no_display },
90   { "Level", 2, 7, display_trg_level },
91   { "Slope", 2, 7, display_trg_slope },
92   { "Delay", 2, 7, display_trg_delay },
93 };
94
95 static int menu_entry;      /* currently selected menu entry */
96
97
98
99
100 /*
101  init_menu
102
103  Description:      This function initializes the menu routines. It sets
104                   the current menu entry to the first entry, indicates the
105                   display is off, and initializes the options (and
106                   hardware) to normal trigger mode, scale displayed, the
107                   fastest sweep rate, a middle trigger level, positive
108                   trigger slope, and minimum delay. Finally, it displays
109                   the menu.
110
111  Arguments:      None.
112  Return Value:   None.
113
114  Input:          None.
115  Output:         The menu is displayed.
116
117  Error Handling:  None.
118
119  Algorithms:     None.
120  Data Structures: None.
121
122  Global Variables: menu_display - reset to FALSE.
123                   menu_entry - reset to first entry (0).
124
125  Author:         Glen George
126  Last Modified:  Mar. 17, 1997
127
128 */
129
130 void init_menu(void)
131 {
132     /* variables */
133     /* none */
134
135
136
137     /* set the menu parameters */
138     menu_entry = 0;      /* first menu entry */
139     menu_display = FALSE; /* menu is not currently displayed (but it will be shortly) */
140
141
142     /* set the scope (option) parameters */
143     set_trigger_mode(NORMAL_TRIGGER); /* normal triggering */
144     set_scale(SCALE_AXES); /* scale is axes */
145     set_sweep(0); /* first sweep rate */
146     set_trg_level((MIN_TRG_LEVEL_SET + MAX_TRG_LEVEL_SET) / 2); /* middle trigger level */
147     set_trg_slope(SLOPE_POSITIVE); /* positive slope */
148     set_trg_delay(MIN_DELAY); /* minimum delay */
149
150

```

```

151     /* now display the menu */
152     display_menu();
153
154
155     /* done initializing, return */
156     return;
157
158 }
159
160
161
162
163 /*
164     clear_menu
165
166     Description:      This function removes the menu from the display. The
167                       trace under the menu is restored. The flag menu_display,
168                       is cleared, indicating the menu is no longer being
169                       displayed. Note: if the menu is not currently being
170                       displayed this function does nothing.
171
172     Arguments:       None.
173     Return Value:    None.
174
175     Input:           None.
176     Output:          The menu if displayed, is removed and the trace under it
177                       is rewritten.
178
179     Error Handling:   None.
180
181     Algorithms:      None.
182     Data Structures: None.
183
184     Global Variables: menu_display - checked and set to FALSE.
185
186     Author:          Glen George
187     Last Modified:   Mar. 13, 1994
188
189 */
190
191 void clear_menu(void)
192 {
193     /* variables */
194     /* none */
195
196
197
198     /* check if the menu is currently being displayed */
199     if (menu_display) {
200
201         /* menu is being displayed - turn it off and restore the trace in that area */
202         restore_menu_trace();
203     }
204
205
206     /* no longer displaying the menu */
207     menu_display = FALSE;
208
209
210     /* all done, return */
211     return;
212 }
213
214
215
216
217
218 /*
219     display_menu
220
221     Description:      This function displays the menu. The trace under the
222                       menu is overwritten (but it was saved). The flag
223                       menu_display, is also set, indicating the menu is
224                       currently being displayed. Note: if the menu is already
225                       being displayed this function does not redisplay it.

```

```

226
227 Arguments:      None.
228 Return Value:   None.
229
230 Input:          None.
231 Output:         The menu is displayed.
232
233 Error Handling:  None.
234
235 Algorithms:     None.
236 Data Structures: None.
237
238 Global Variables: menu_display - set to TRUE.
239                  menu_entry  - used to highlight currently selected entry.
240
241 Author:         Glen George
242 Last Modified:  Mar. 13, 1994
243
244 */
245
246 void display_menu(void)
247 {
248     /* variables */
249     int i;        /* loop index */
250
251
252
253     /* check if the menu is currently being displayed */
254     if (!menu_display) {
255
256         /* menu is not being displayed - turn it on */
257         /* display it entry by entry */
258         for (i = 0; i < NO_MENU_ENTRIES; i++) {
259
260             /* display this entry - check if it should be highlighted */
261             if (i == menu_entry)
262                 /* currently selected entry - highlight it */
263                 display_entry(i, TRUE);
264             else
265                 /* not the currently selected entry - "normal video" */
266                 display_entry(i, FALSE);
267         }
268     }
269
270
271     /* now are displaying the menu */
272     menu_display = TRUE;
273
274
275     /* all done, return */
276     return;
277 }
278
279
280
281
282
283 /*
284 refresh_menu
285
286 Description:      This function displays the menu if it is currently being
287                  displayed. The trace under the menu is overwritten (but
288                  it was already saved).
289
290 Arguments:      None.
291 Return Value:   None.
292
293 Input:          None.
294 Output:         The menu is displayed.
295
296 Error Handling:  None.
297
298 Algorithms:     None.
299 Data Structures: None.
300

```

```

301 Global Variables: menu_display - determines if menu should be displayed.
302
303 Author: Glen George
304 Last Modified: Mar. 8, 1994
305
306 */
307
308 void refresh_menu(void)
309 {
310     /* variables */
311     /* none */
312
313
314
315     /* check if the menu is currently being displayed */
316     if (menu_display) {
317
318         /* menu is currently being displayed - need to refresh it */
319         /* do this by turning off the display, then forcing it back on */
320         menu_display = FALSE;
321         display_menu();
322     }
323
324
325     /* refreshed the menu if it was displayed, now return */
326     return;
327 }
328
329
330
331
332
333 /*
334 reset_menu
335
336 Description: This function resets the current menu selection to the
337             first menu entry. If the menu is currently being
338             displayed the display is updated.
339
340 Arguments: None.
341 Return Value: None.
342
343 Input: None.
344 Output: The menu display is updated if it is being displayed.
345
346 Error Handling: None.
347
348 Algorithms: None.
349 Data Structures: None.
350
351 Global Variables: menu_display - checked to see if menu is displayed.
352                  menu_entry - reset to 0 (first entry).
353
354 Author: Glen George
355 Last Modified: Mar. 17, 1997
356
357 */
358
359 void reset_menu(void)
360 {
361     /* variables */
362     /* none */
363
364
365
366     /* check if the menu is currently being displayed */
367     if (menu_display) {
368
369         /* menu is being displayed */
370         /* remove highlight from currently selected entry */
371         display_entry(menu_entry, FALSE);
372     }
373
374
375     /* reset the currently selected entry */

```

```

376     menu_entry = 0;
377
378
379     /* finally, highlight the first entry if the menu is being displayed */
380     if (menu_display)
381         display_entry(menu_entry, TRUE);
382
383
384
385     /* all done, return */
386     return;
387
388 }
389
390
391
392
393 /*
394 next_entry
395
396 Description:      This function changes the current menu selection to the
397                   next menu entry.  If the current selection is the last
398                   entry in the menu, it is not changed.  If the menu is
399                   currently being displayed, the display is updated.
400
401 Arguments:        None.
402 Return Value:     None.
403
404 Input:            None.
405 Output:           The menu display is updated if it is being displayed and
406                   the entry selected changes.
407
408 Error Handling:    None.
409
410 Algorithms:       None.
411 Data Structures:  None.
412
413 Global Variables: menu_display - checked to see if menu is displayed.
414                   menu_entry   - updated to a new entry (if not at end).
415
416 Author:           Glen George
417 Last Modified:    Mar. 13, 1994
418
419 */
420
421 void next_entry(void)
422 {
423     /* variables */
424     /* none */
425
426
427
428     /* only update if not at end of the menu */
429     if (menu_entry < (NO_MENU_ENTRIES - 1)) {
430
431         /* not at the end of the menu */
432
433         /* turn off current entry if displaying */
434         if (menu_display)
435             /* displaying menu - turn off currently selected entry */
436             display_entry(menu_entry, FALSE);
437
438         /* update the menu entry to the next one */
439         menu_entry++;
440
441         /* now highlight this entry if displaying the menu */
442         if (menu_display)
443             /* displaying menu - highlight newly selected entry */
444             display_entry(menu_entry, TRUE);
445     }
446
447
448     /* all done, return */
449     return;
450

```

```

451 }
452
453
454
455
456 /*
457     previous_entry
458
459     Description:      This function changes the current menu selection to the
460                       previous menu entry.  If the current selection is the
461                       first entry in the menu, it is not changed.  If the menu
462                       is currently being displayed, the display is updated.
463
464     Arguments:        None.
465     Return Value:      None.
466
467     Input:             None.
468     Output:            The menu display is updated if it is being displayed and
469                       the currently selected entry changes.
470
471     Error Handling:    None.
472
473     Algorithms:        None.
474     Data Structures:   None.
475
476     Global Variables:  menu_display - checked to see if menu is displayed.
477                       menu_entry - updated to a new entry (if not at start).
478
479     Author:            Glen George
480     Last Modified:     Mar. 13, 1994
481 */
482
483 void previous_entry(void)
484 {
485     /* variables */
486     /* none */
487
488
489
490
491     /* only update if not at the start of the menu */
492     if (menu_entry > 0) {
493
494         /* not at the start of the menu */
495
496         /* turn off current entry if displaying */
497         if (menu_display)
498             /* displaying menu - turn off currently selected entry */
499             display_entry(menu_entry, FALSE);
500
501         /* update the menu entry to the previous one */
502         menu_entry--;
503
504         /* now highlight this entry if displaying the menu */
505         if (menu_display)
506             /* displaying menu - highlight newly selected entry */
507             display_entry(menu_entry, TRUE);
508     }
509
510
511
512     /* all done, return */
513     return;
514 }
515
516
517
518
519
520 /*
521     menu_entry_left
522
523     Description:      This function handles the <Left> key for the current menu
524                       selection.  It does this by doing a table lookup on the
525                       current menu selection.

```

```

526 Arguments:      None.
527 Return Value:   None.
528
529
530 Input:          None.
531 Output:         The menu display is updated if it is being displayed and
532                 the <Left> key causes a change to the display.
533
534 Error Handling:  None.
535
536 Algorithms:     Table lookup is used to determine what to do for the
537                 input key.
538 Data Structures: An array holds the table of key processing routines.
539
540 Global Variables: menu_entry - used to select the processing function.
541
542 Author:         Glen George
543 Last Modified:  May 9, 2006
544
545 */
546
547 void menu_entry_left(void)
548 {
549     /* variables */
550
551     /* key processing functions */
552     static void (* const process[])(void) =
553         /* Mode      Scale      Sweep      Trigger      */
554         { mode_down,  scale_down,  sweep_down,  trace_rearm,
555           trg_level_down, trg_slope_toggle, trg_delay_down, };
556         /* Level      Slope      Delay      */
557
558
559
560     /* invoke the appropriate <Left> key function */
561     process[menu_entry]();
562
563     /* if displaying menu entries, display the new value */
564     /* note: since it is being changed - know this option is selected */
565     if (menu_display) {
566         menu[menu_entry].display((MENU_X + menu[menu_entry].opt_off),
567                                 (MENU_Y + menu_entry), OPTION_SELECTED);
568     }
569
570
571     /* all done, return */
572     return;
573 }
574
575
576
577
578
579 /*
580 menu_entry_right
581
582 Description:     This function handles the <Right> key for the current
583                 menu selection. It does this by doing a table lookup on
584                 the current menu selection.
585
586 Arguments:      None.
587 Return Value:   None.
588
589 Input:          None.
590 Output:         The menu display is updated if it is being displayed and
591                 the <Right> key causes a change to the display.
592
593 Error Handling:  None.
594
595 Algorithms:     Table lookup is used to determine what to do for the
596                 input key.
597 Data Structures: An array holds the table of key processing routines.
598
599 Global Variables: menu      - used to display the new menu value.
600                 menu_entry - used to select the processing function.

```



```

601
602 Author:      Glen George
603 Last Modified: May 9, 2006
604
605 */
606
607 void menu_entry_right(void)
608 {
609     /* variables */
610
611     /* key processing functions */
612     static void (* const process[])(void) =
613         /* Mode      Scale      Sweep      Trigger      */
614         { mode_up,    scale_up,    sweep_up,    trace_rearm,
615           trg_level_up, trg_slope_toggle, trg_delay_up,
616           /* Level      Slope      Delay      */
617         };
618
619
620     /* invoke the appropriate <Right> key function */
621     process[menu_entry]();
622
623     /* if displaying menu entries, display the new value */
624     /* note: since it is being changed - know this option is selected */
625     if (menu_display) {
626         menu[menu_entry].display((MENU_X + menu[menu_entry].opt_off),
627                                 (MENU_Y + menu_entry), OPTION_SELECTED);
628     }
629
630
631     /* all done, return */
632     return;
633 }
634
635
636
637
638
639 /*
640 display_entry
641
642 Description:      This function displays the passed menu entry and its
643                   current option setting. If the second argument is TRUE
644                   it displays them with color SELECTED and OPTION_SELECTED
645                   respectively. If the second argument is FALSE it
646                   displays the menu entry with color NORMAL and the option
647                   setting with color OPTION_NORMAL.
648
649 Arguments:        entry (int)      - menu entry to be displayed.
650                   selected (int) - whether or not the menu entry is
651                                   currently selected (determines the color
652                                   with which the entry is output).
653 Return Value:      None.
654
655 Input:             None.
656 Output:            The menu entry is output to the LCD.
657
658 Error Handling:     None.
659
660 Algorithms:        None.
661 Data Structures:    None.
662
663 Global Variables:  menu - used to display the menu entry.
664
665 Author:            Glen George
666 Last Modified:      Aug. 13, 2004
667
668 */
669
670 static void display_entry(int entry, int selected)
671 {
672     /* variables */
673     /* none */
674
675

```

```
676
677 /* output the menu entry with the appropriate color */
678 plot_string((MENU_X + menu[entry].h_off), (MENU_Y + entry), menu[entry].s,
679             (selected ? SELECTED : NORMAL));
680 /* also output the menu option with the appropriate color */
681 menu[entry].display((MENU_X + menu[entry].opt_off), (MENU_Y + entry),
682                    (selected ? OPTION_SELECTED : OPTION_NORMAL));
683
684
685 /* all done outputting this menu entry - return */
686 return;
687
688 }
689
```