

```

1  /*****
2  /*
3  /*                                KEYPROC                                */
4  /*                                Key Processing Functions                    */
5  /*                                Digital Oscilloscope Project                */
6  /*                                EE/CS 52                                    */
7  /*
8  /*****
9
10 /*
11 This file contains the key processing functions for the Digital
12 Oscilloscope project. These functions are called by the main loop of the
13 system. The functions included are:
14     menu_down - process the <Down> key while in a menu
15     menu_key  - process the <Menu> key
16     menu_left - process the <Left> key while in a menu
17     menu_right - process the <Right> key while in a menu
18     menu_up   - process the <Up> key while in a menu
19     no_action - nothing to do
20
21 The local functions included are:
22     none
23
24 The locally global variable definitions included are:
25     none
26
27
28 Revision History
29     3/8/94   Glen George           Initial revision.
30     3/13/94  Glen George           Updated comments.
31 */
32
33
34
35 /* library include files */
36 /* none */
37
38 /* local include files */
39 #include "scopedef.h"
40 #include "keyproc.h"
41 #include "menu.h"
42
43
44
45
46 /*
47     no_action
48
49 Description:      This function handles a key when there is nothing to be
50                   done. It just returns.
51
52 Arguments:        cur_state (enum status) - the current system state.
53 Return Value:      (enum status) - the new system state (same as current
54                   state).
55
56 Input:            None.
57 Output:            None.
58
59 Error Handling:    None.
60
61 Algorithms:        None.
62 Data Structures:   None.
63
64 Global Variables:  None.
65
66 Author:            Glen George
67 Last Modified:     Mar. 8, 1994
68
69 */
70
71 enum status no_action(enum status cur_state)
72 {
73     /* variables */
74     /* none */
75

```

```

76
77
78     /* return the current state */
79     return  cur_state;
80
81 }
82
83
84
85
86 /*
87     menu_key
88
89     Description:      This function handles the <Menu> key.  If the passed
90                       state is MENU_ON, the menu is turned off.  If the passed
91                       state is MENU_OFF, the menu is turned on.  The returned
92                       state is the "opposite" of the passed state.
93
94     Arguments:      cur_state (enum status) - the current system state.
95     Return Value:    (enum status) - the new system state ("opposite" of the
96                       as current state).
97
98     Input:           None.
99     Output:          The menu is either turned on or off.
100
101     Error Handling:  None.
102
103     Algorithms:      None.
104     Data Structures: None.
105
106     Global Variables: None.
107
108     Author:          Glen George
109     Last Modified:    Mar. 8, 1994
110
111 */
112
113 enum status  menu_key(enum status cur_state)
114 {
115     /* variables */
116     /* none */
117
118
119
120     /* check if need to turn the menu on or off */
121     if (cur_state == MENU_ON)
122         /* currently the menu is on, turn it off */
123         clear_menu();
124     else
125         /* currently the menu is off, turn it on */
126         display_menu();
127
128
129     /* all done, return the "opposite" of the current state */
130     if (cur_state == MENU_ON)
131         /* state was MENU_ON, change it to MENU_OFF */
132         return  MENU_OFF;
133     else
134         /* state was MENU_OFF, change it to MENU_ON */
135         return  MENU_ON;
136
137 }
138
139
140
141
142 /*
143     menu_up
144
145     Description:      This function handles the <Up> key when in a menu.  It
146                       goes to the previous menu entry and leaves the system
147                       state unchanged.
148
149     Arguments:      cur_state (enum status) - the current system state.
150     Return Value:    (enum status) - the new system state (same as current

```

```

151         state).
152
153     Input:          None.
154     Output:         The menu display is updated.
155
156     Error Handling:  None.
157
158     Algorithms:      None.
159     Data Structures: None.
160
161     Global Variables: None.
162
163     Author:          Glen George
164     Last Modified:    Mar. 8, 1994
165
166 */
167
168 enum status  menu_up(enum status cur_state)
169 {
170     /* variables */
171     /* none */
172
173
174
175     /* go to the previous menu entry */
176     previous_entry();
177
178
179     /* return the current state */
180     return  cur_state;
181 }
182
183
184
185
186
187 /*
188     menu_down
189
190     Description:      This function handles the <Down> key when in a menu.  It
191                       goes to the next menu entry and leaves the system state
192                       unchanged.
193
194     Arguments:        cur_state (enum status) - the current system state.
195     Return Value:      (enum status) - the new system state (same as current
196                       state).
197
198     Input:            None.
199     Output:           The menu display is updated.
200
201     Error Handling:    None.
202
203     Algorithms:        None.
204     Data Structures:   None.
205
206     Global Variables:  None.
207
208     Author:           Glen George
209     Last Modified:     Mar. 8, 1994
210
211 */
212
213 enum status  menu_down(enum status cur_state)
214 {
215     /* variables */
216     /* none */
217
218
219
220     /* go to the next menu entry */
221     next_entry();
222
223
224     /* return the current state */
225     return  cur_state;

```

```

226 }
227 }
228
229
230
231
232 /*
233     menu_left
234
235     Description:      This function handles the <Left> key when in a menu. It
236                       invokes the left function for the current menu entry and
237                       leaves the system state unchanged.
238
239     Arguments:        cur_state (enum status) - the current system state.
240     Return Value:     (enum status) - the new system state (same as current
241                       state).
242
243     Input:            None.
244     Output:           The menu display may be updated.
245
246     Error Handling:   None.
247
248     Algorithms:       None.
249     Data Structures:  None.
250
251     Global Variables: None.
252
253     Author:           Glen George
254     Last Modified:    Mar. 8, 1994
255 */
256
257 enum status menu_left(enum status cur_state)
258 {
259     /* variables */
260     /* none */
261
262
263
264
265     /* invoke the <Left> key function for the current menu entry */
266     menu_entry_left();
267
268
269     /* return the current state */
270     return cur_state;
271 }
272
273
274
275
276
277 /*
278     menu_right
279
280     Description:      This function handles the <Right> key when in a menu. It
281                       invokes the right function for the current menu entry and
282                       leaves the system state unchanged.
283
284     Arguments:        cur_state (enum status) - the current system state.
285     Return Value:     (enum status) - the new system state (same as current
286                       state).
287
288     Input:            None.
289     Output:           The menu display may be updated.
290
291     Error Handling:   None.
292
293     Algorithms:       None.
294     Data Structures:  None.
295
296     Global Variables: None.
297
298     Author:           Glen George
299     Last Modified:    Mar. 8, 1994
300

```

```
301 */
302
303 enum status  menu_right(enum status cur_state)
304 {
305     /* variables */
306     /* none */
307
308
309
310     /* invoke the <Right> key function for the current menu entry */
311     menu_entry_right();
312
313
314     /* return the current state */
315     return  cur_state;
316
317 }
318
```