D:\Atwin32.dev\Samples\UIEX\UIEX3

```
1  *
2  ************************************************************************
3  * Copyright (c) 2004 Zumasys, Inc. as an unpublished work. Permission is *
4  * hereby granted, free of charge, to any person obtaining a copy of this *
5  * software, to use the software without restriction, including without   *
6  * limitation the rights to use, copy, modify, merge, publish or          *
7  * distribute the software, and to permit persons to whom the software is *
8  * furnished to do so, subject to the following conditions: This          *
9  * copyright notice and permission notice shall be included in all copies *
10 * or substantial portions of the software. THE SOFTWARE IS PROVIDED "AS  *
11 * IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT   *
12 * NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A        *
13 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE          *
14 * COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, *
15 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT *
16 * OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  *
17 * THE SOFTWARE.                                                          *
18 ************************************************************************
19 ************************************************************************
20 *
21 * USER INTERFACE EXAMPLE PROGRAM 3
22 *
23 ************************************************************************
24 ************************************************************************
25 *
26 * This example illustrates a character-based interface which uses AccuTerm
27 * Visual Styles to display screen elements using a Windows-like look (if
28 * available). In this example, Visual Styles (border effects and colors)
29 * are associated with certain display attributes. The program will run
30 * without Visual Styles if used with dumb terminals, older versions of
31 * AccuTerm or other terminal emulators. This example incorporates AccuTerm
32 * Imaging to display a picture associated with each customer record (if
33 * available).
34 *
35 ************************************************************************
36 *
37 * The CUST.REC INCLUDE item declares the CUST.REC array, which is used to
38 * store a working copy of the current data record. It also defines the
39 * record layout by equating field names to array positions in the CUST.REC
40 * array.
41 *
42 $INCLUDE UIEX.CUST.REC
43 *
44 ************************************************************************
45 *
```

D:\Atwin32.dev\Samples\UIEX\UIEX4

```
1  *
2  ************************************************************************
3  * Copyright (c) 2004 Zumasys, Inc. as an unpublished work. Permission is *
4  * hereby granted, free of charge, to any person obtaining a copy of this *
5  * software, to use the software without restriction, including without   *
6  * limitation the rights to use, copy, modify, merge, publish or          *
7  * distribute the software, and to permit persons to whom the software is *
8  * furnished to do so, subject to the following conditions: This          *
9  * copyright notice and permission notice shall be included in all copies *
10 * or substantial portions of the software. THE SOFTWARE IS PROVIDED "AS  *
11 * IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT   *
12 * NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A        *
13 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE          *
14 * COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, *
15 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT *
16 * OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  *
17 * THE SOFTWARE.                                                          *
18 ************************************************************************
19 ************************************************************************
20 *
21 * USER INTERFACE EXAMPLE PROGRAM 4
22 *
23 ************************************************************************
24 ************************************************************************
25 *
26 * This example illustrates a character-based interface using the Zumasys
27 * Smart User Interface library for data entry. Most display and input
28 * processing is handled by the SUI subroutines, rather than using PRINT
29 * and INPUT statements. By using the SUI library, input fields accept
30 * cursor keys, editing keys, function keys and mouse clicks. The SUI
31 * subroutines also utilize AccuTerm Visual Styles to display screen
32 * elements using a Windows-like look (if avaliable).
33 *
34 ************************************************************************
35 *
36 * To use the SUI library, you must include SUI.TERMDEF. This INCLUDE item
37 * contains EQUates for various terminal functions like visual attributes
38 * and line graphics, as well as constants for the keyboard commands like
39 * TERM.ENTER$ or TERM.PGUP$.
40 *
41 $INCLUDE SUIBP SUI.TERMDEF
42 *
43 ************************************************************************
44 *
45 * The CUST.REC INCLUDE item declares the CUST.REC array, which is used to
46 * store a working copy of the current data record. It also defines the
47 * record layout by equating field names to array positions in the CUST.REC
48 * array.
49 *
50 $INCLUDE UIEX.CUST.REC
51 *
52 ************************************************************************
53 *
```

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines				
Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX3

```
46 * EQUates for control characters and delimiters and other constants
47 *
48 EQU VM TO CHAR(253)
49 EQU BEL TO CHAR(7)
50 EQU ESC TO CHAR(27)
51 EQU STX TO CHAR(2)
52 EQU CR TO CHAR(13)
53 EQU FALSE TO 0
54 EQU TRUE TO 1
55 *
56 ************************************************************************
57 *
58 * This program uses AccuTerm Imaging to display pictures associated with
59 * records in the customer file. This EQUate defines the path where the
60 * image files are located.
61 *
62 $INCLUDE UIEX.PIX.PATH
63 *
64 * Check for image support (no images for dumb terminal or AccuTerm Lite)
65 *
66 CALL FTVSINF('','','',CAPABILITIES,'','','','')
67 IF INDEX(CAPABILITIES,'I',1) THEN IMGFLG = TRUE ELSE IMGFLG = FALSE
68 *
69 ************************************************************************
70 *
71 * Open our files...
72 *
73 OPEN 'CUST.SAMPLE' TO FN.CUST ELSE PRINT 'NO CUST.SAMPLE FILE';STOP
74 OPEN 'CUST.SAMPLE.CTRL' TO FN.CUST.CTRL ELSE PRINT 'NO CUST.SAMPLE.CTRL FILE
75 OPEN 'CUST.SAMPLE.XREF' TO FN.CUST.XREF ELSE PRINT 'NO CUST.SAMPLE.XREF';STO
76 *
77 ************************************************************************
78 *
79 * Define the data field control structures. NUMFLDS is the number of
80 * fields. The CONTROL array defines the field control elements such as
```

D:\Atwin32.dev\Samples\UIEX\UIEX4

```
54 * EQUates for control characters and delimiters and other constants
55 *
56 EQU VM TO CHAR(253)
57 EQU BEL TO CHAR(7)
58 EQU ESC TO CHAR(27)
59 EQU STX TO CHAR(2)
60 EQU CR TO CHAR(13)
61 EQU FALSE TO 0
62 EQU TRUE TO 1
63 *
64 ************************************************************************
65 *
66 * Any routine that uses the SUI library must call SUI.GET.TERM to
67 * initialize the TermDef array. The TermDef array is a required argument
68 * for all of the SUI subroutines. It is much better to call this routine
69 * one time and pass the TermDef array between various routines instead of
70 * calling it in each application subroutine that uses SUI.
71 *
72 CALL SUI.GET.TERM(MAT TermDef)
73 *
74 ************************************************************************
75 *
76 * Once the TermDef array is initialized by SUI.GET.TERM, the TERM.RESET
77 * string is sent to the terminal to initialize the terminal state. The
78 * reset string programs necessary function keys and initalizes any other
79 * terminal state as required for each particular terminal type.
80 *
81 PRINT TERM.RESET:
82 *
83 ************************************************************************
84 *
85 * This program uses AccuTerm Imaging to display pictures associated with
86 * records in the customer file. This EQUate defines the path where the
87 * image files are located.
88 *
89 $INCLUDE UIEX.PIX.PATH
90 *
91 * Check for image support (no images if dumb terminal or AccuTerm Lite)
92 *
93 IF INDEX(TERM.SPECIAL,'I',1) THEN IMGFLG = TRUE ELSE IMGFLG = FALSE
94 *
95 ************************************************************************
96 *
97 * Open our files...
98 *
99 OPEN 'CUST.SAMPLE' TO FN.CUST ELSE PRINT 'NO CUST.SAMPLE FILE'; STOP
100 OPEN 'CUST.SAMPLE.CTRL' TO FN.CUST.CTRL ELSE PRINT 'NO CUST.SAMPLE.CTRL FILE
101 OPEN 'CUST.SAMPLE.XREF' TO FN.CUST.XREF ELSE PRINT 'NO CUST.SAMPLE.XREF FILE
102 *
103 ************************************************************************
104 *
105 * Define the data field control structures. NUMFLDS is the number of
106 * fields. The CONTROL array defines the field control elements such as
```

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines				
Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX3

```
81 * field label, label position, field position and size, and field prompt.
82 * The IDATA array stores the current field values, and is initialized from
83 * the CUST.REC array when a data record is read from the file. When the
84 * record is updated, values are copied from the IDATA array to the
85 * CUST.REC array and the CUST.REC array is passed to the CUST.UPDATE
86 * subroutine to update the data and index files (a separate subroutine is
87 * used to update the file since the update process may handle other
88 * functions like updating indexes).
89 *
90 NUMFLDS = 12
91 DIM CONTROL(12)
92 DIM IDATA(12)
93 *
94 * Define field control elements
95 *  value 1: field label text
96 *  value 2: field label column
97 *  value 3: field label row
98 *  value 4: field label width (0 for actual width, no padding)
99 *  value 5: field data column
100 *  value 6: field data row
101 *  value 7: field data width
102 *  value 8: field data height
103 *  value 9: field prompt message
104 CONTROL(1) = 'Customer ID???0?8??0?ýEnter the ID, or name to search for, or
105 CONTROL(2) = 'Contact???0?8??0?ýEnter the contact name'
106 CONTROL(3) = 'Company name???0?8??0?ýEnter the company name'
107 CONTROL(4) = 'Address line 1???0?8??0?ýEnter address line 1'
108 CONTROL(5) = 'Address line 2???0?8??0?ýEnter address line 2'
109 CONTROL(6) = 'City???0?8??5?ýEnter the city'
110 CONTROL(7) = 'State or province???0?8??5?ýEnter the state or province abbrev
111 CONTROL(8) = 'Zip/postal code??0?0?8?0?0?ýEnter the zip or postal code'
112 CONTROL(9) = 'Country??1?0?8?1?8?ýEnter the country'
113 CONTROL(10) = 'Phone???2?0?8?2?5?ýEnter the phone number'
114 CONTROL(11) = 'Fax??3?0?8?3?5?ýEnter the fax number'
115 CONTROL(12) = 'Notes??4?0?8?4?0?ýEnter any notes about this customer'
116 *
117 ************************************************************************
118 *
119 * Define some screen control strings for prompts & errors
120 *
121 PROMPT ''
122 CLR = @(-1)         ;* Clear entire screen
123 CEOL = @(-4)        ;* Clear to end of line
124 PL = @(5,22):CEOL   ;* Prompt line
125 EL = @(5,23):CEOL   ;* Error line
126 *
127 * Wyse 60 attributes for NORMAL, REVERSE, DIM REVERSE and UNDERLINE
128 * REVERSE (we like Wyse 60 because the attributes dont take any space on
129 * the screen, and more than one attribute can be displayed at one time).
130 * If running AccuTerm in Viewpoint A2 Enhanced emulation, you can use the
131 * ADDS 4000 attributes instead. Just change the upper-case "G" below to
132 * lower-case "g".
133 *
134 NORMAL = ESC:'G0'   ;* Normal - display headings & field labels
```

D:\Atwin32.dev\Samples\UIEX\UIEX4

```
107 * field label, label position, field position and size, and field prompt.
108 * The IDATA array stores the current field values, and is initialized
109 * from the CUST.REC array when a data record is read from the file. When
110 * the record is updated, values are copied from the IDATA array to the
111 * CUST.REC array and the CUST.REC array is passed to the CUST.UPDATE
112 * subroutine to update the data and index files (a separate subroutine is
113 * used to update the file since the update process may handle other
114 * functions like updating indexes).
115 *
116 NUMFLDS = 12
117 DIM CONTROL(12)
118 DIM IDATA(12)
119 *
120 * Define field control elements
121 *  value 1: field label text
122 *  value 2: field label column
123 *  value 3: field label row
124 *  value 4: field label width (0 for actual width, no padding)
125 *  value 5: field data column
126 *  value 6: field data row
127 *  value 7: field data width
128 *  value 8: field data height
129 *  value 9: field prompt message
130 CONTROL(1) = 'Customer ID???0?8??0?ýEnter the ID, or name to search for, or
131 CONTROL(2) = 'Contact???0?8??0?ýEnter the contact name'
132 CONTROL(3) = 'Company name???0?8??0?ýEnter the company name'
133 CONTROL(4) = 'Address line 1???0?8??0?ýEnter address line 1'
134 CONTROL(5) = 'Address line 2???0?8??0?ýEnter address line 2'
135 CONTROL(6) = 'City???0?8??5?ýEnter the city'
136 CONTROL(7) = 'State or province???0?8??5?ýEnter the state or province abbrev
137 CONTROL(8) = 'Zip/postal code??0?0?8?0?0?ýEnter the zip or postal code'
138 CONTROL(9) = 'Country??1?0?8?1?8?ýEnter the country'
139 CONTROL(10) = 'Phone???2?0?8?2?5?ýEnter the phone number'
140 CONTROL(11) = 'Fax??3?0?8?3?5?ýEnter the fax number'
141 CONTROL(12) = 'Notes??4?0?8?4?0?ýEnter any notes about this customer'
142 *
143 ************************************************************************
144 *
145 * Define some screen control strings for prompts & errors
146 *
147 PROMPT ''
148 PL = @(5,22):TERM.CEOL
149 EL = @(5,23):TERM.CEOL
150 *
```

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines

Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX3

```
135 REVERSE = ESC:'G4' ;* Reverse - display active field data
136 DIMREV = ESC:'Gt'  ;* Dim Reverse - display inactive field data
137 UNDREV = ESC:'G<'  ;* Underline Reverse - display warnings
138 *
139 ************************************************************************
140 *
141 * This program processes one customer record at a time, and is organized
142 * using three nested loops. The outermost loop (the RECORD loop) is
143 * executed once for each record accessed. The loop repeats until the XIT
144 * control variable is set to TRUE.
145 *
146 XIT = FALSE
147 LOOP UNTIL XIT DO
148  *
149  ************************************************************************
150  *
151  * Before prompting for the customer ID, reset the internal data array
152  * (IDATA) and CUST.ID variables, then clear the screen and display the
153  * heading and field labels.
154  *
155  GOSUB RESTART
156  GOSUB DSPSCRN
157  *
158  ************************************************************************
159  *
160  * The middle loop is the ACTION loop. It is executed for the current
161  * record until the user performs an action that terminates processing of
162  * that record, such as exiting, cancelling, saving or deleting the
163  * record. The field number to begin prompting (NXTFLD) is initialized to
164  * 1, causing the ID field to be prompted first. The loop immediately
165  * enters the PROMPT loop, followed by the field modification prompt. The
166  * loop repeats until the DONE control variable is set to TRUE.
167  *
168  NXTFLD = 1
169  DONE = FALSE
170  LOOP
171   *
172   ************************************************************************
173   *
174   * The inner loop is the PROMPT loop. It is executed for each prompt
175   * field as specified by the NXTFLD variable. The prompt loop simply
176   * calls the local INPUT.FIELD subroutine which performs field input,
177   * data validation and keyboard command decoding. The FIELD loop repeats
178   * until the field number is greater than the number of fields, or until
179   * the DONE control variable is set to TRUE. The DONE control variable
180   * may be set to TRUE in the INPUT.FIELD subroutine, if the user enters a
181   * NULL to for the item-ID.
182   *
183   LOOP
184    CURFLD = NXTFLD
185   UNTIL DONE OR CURFLD > NUMFLDS DO
186    *
187    ************************************************************************
188    *
```

D:\Atwin32.dev\Samples\UIEX\UIEX4

```
151 ************************************************************************
152 *
153 * This program processes one customer record at a time, and is organized
154 * using three nested loops. The outermost loop (the RECORD loop) is
155 * executed once for each record accessed. The loop repeats until the XIT
156 * control variable is set to TRUE.
157 *
158 XIT = FALSE
159 LOOP UNTIL XIT DO
160  *
161  ************************************************************************
162  *
163  * Before prompting for the customer ID, reset the internal data array
164  * (IDATA) and CUST.ID variables, then clear the screen and display the
165  * heading and field labels.
166  *
167  GOSUB RESTART
168  GOSUB DSPSCRN
169  *
170  ************************************************************************
171  *
172  * The middle loop is the ACTION loop. It is executed for the current
173  * record until the user performs an action that terminates processing of
174  * that record, such as exiting, cancelling, saving or deleting the
175  * record. The field number to begin prompting (NXTFLD) is initialized to
176  * 1, causing the ID field to be prompted first. The loop immediately
177  * enters the PROMPT loop, followed by the field modification prompt. The
178  * loop repeats until the DONE control variable is set to TRUE.
179  *
180  NXTFLD = 1
181  DONE = FALSE
182  LOOP
183   *
184   ************************************************************************
185   *
186   * The inner loop is the PROMPT loop. It is executed for each prompt
187   * field as specified by the NXTFLD variable. The prompt loop simply
188   * calls the local INPUT.FIELD subroutine which performs field input,
189   * data validation and keyboard command decoding. The FIELD loop repeats
190   * until the field number is greater than the number of fields, or until
191   * the DONE control variable is set to TRUE. The DONE control variable
192   * may be set to TRUE in the INPUT.FIELD subroutine, if the user enters a
193   * NULL to for the item-ID.
194   *
195   LOOP
196    CURFLD = NXTFLD
197   UNTIL DONE OR CURFLD > NUMFLDS DO
198    *
199    ************************************************************************
200    *
```

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines				
Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX3

```
189    * Prompt for the next field. Update the NXTFLD variable with the field
190    * number to prompt next.
191    *
192    GOSUB INPUT.FIELD
193   REPEAT ;* end of PROMPT loop
194   *
195   ************************************************************************
196   *
197   * If the FIELD loop exited with the DONE control variable set to TRUE,
198   * bypass the modification prompt because no action is required.
199   * Otherwise, prompt for which field to modify, or other actions such as
200   * save or delete.
201   *
202   IF NOT(DONE) THEN
203    *
204    NXTFLD = NUMFLDS + 1 ;* assume we need to reprompt for action
205    *
206    PRINT PL:'Enter field number to modify or FI to save, DE to delete, EX to
207    INPUT ANS:
208    PRINT EL:
209    *
210    ************************************************************************
211    *
212    * Decode the response
213    *
214    ANS = OCONV(ANS, 'MCU')
215    BEGIN CASE
216     CASE NUM(ANS) AND ANS >= 1 AND ANS <= NUMFLDS; NXTFLD = ANS
217     CASE ANS EQ 'FI'; GOSUB SAVE.RECORD
218     CASE ANS EQ 'DE'; GOSUB DELETE.RECORD
219     CASE ANS EQ 'EX' OR ANS EQ ''; GOSUB CHECK.EXIT
220    END CASE
221    *
222   END
223   *
224  UNTIL DONE DO REPEAT ;* end of ACTION loop
225  *
226 REPEAT ;* end of RECORD loop
227 *
228 ************************************************************************
229 *
230 * All done - clear the screen and exit!
231 PRINT CLR:
232 STOP
233 *
234 *
235 ************************************************************************
236 ************************************************************************
237 * LOCAL SUBROUTINES
238 ************************************************************************
239 ************************************************************************
240 *
241 *
```

D:\Atwin32.dev\Samples\UIEX\UIEX4

```
201    * Prompt for the next field. Update the NXTFLD variable with the field
202    * number to prompt next, taking into account cursor keys and mouse
203    * clicks.
204    *
205    GOSUB INPUT.FIELD
206   REPEAT ;* end of PROMPT loop
207   *
208   ************************************************************************
209   *
210   * If the FIELD loop exited with the DONE control variable set to TRUE,
211   * bypass the modification prompt because no action is required.
212   * Otherwise, prompt for which field to modify, or other actions such as
213   * save or delete.
214   *
215   IF NOT(DONE) THEN
216    *
217    NXTFLD = NUMFLDS + 1 ;* assume we need to reprompt for action
218    *
219    PRINT PL:'Enter field number to modify or FI to save, DE to delete, EX to
220    INPUT ANS:
221    PRINT EL:
222    *
223    ************************************************************************
224    *
225    * Decode the response
226    *
227    ANS = OCONV(ANS, 'MCU')
228    BEGIN CASE
229     CASE NUM(ANS) AND ANS >= 1 AND ANS <= NUMFLDS; NXTFLD = ANS
230     CASE ANS EQ 'FI'; GOSUB SAVE.RECORD
231     CASE ANS EQ 'DE'; GOSUB DELETE.RECORD
232     CASE ANS EQ 'EX' OR ANS EQ ''; GOSUB CHECK.EXIT
233    END CASE
234    *
235   END
236   *
237  UNTIL DONE DO REPEAT ;* end of ACTION loop
238  *
239 REPEAT ;* end of RECORD loop
240 *
241 ************************************************************************
242 *
243 * All done - clear the screen, restore the cursor and exit!
244 PRINT TERM.CLEAR:TERM.CSRON:
245 STOP
246 *
247 *
248 ************************************************************************
249 ************************************************************************
250 * LOCAL SUBROUTINES
251 ************************************************************************
252 ************************************************************************
253 *
254 *
```

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines				
Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX3

```
242 ************************************************************************
243 *
244 * The INPUT.FIELD subroutine is the main prompting routine. This routine
245 * displays the prompt string (from the CONTROL array), and enters a loop,
246 * prompting for a specified field (CURFLD) and setting the next field
247 * variable (NXTFLD) appropriately. The loop repeats until the NXTFLD
248 * (initially set to CURFLD) changes. This causes the field prompt to be
249 * repeated in case invalid data is entered (illegal customer ID, etc.) If
250 * a NULL is entered for the ID field, and there is no current record, the
251 * ACTION loop control variable, DONE, and the RECORD loop control
252 * variable, XIT, are set to TRUE, and this routine exits.
253 *
254 INPUT.FIELD:
255 *
256 ************************************************************************
257 *
258 * Display the prompt message
259 *
260 PRINT PL:CONTROL(CURFLD)<1,9>:
261 *
262 ************************************************************************
263 *
264 * Initialize current value, next field
265 *
266 PREVAL = IDATA(CURFLD) ;* Save previous field value to detect change in valu
267 *
268 ************************************************************************
269 *
270 * Prompt for this field until NXTFLD variable is updated
271 *
272 LOOP
273  *
274  ************************************************************************
275  *
276  * Assume next field number is next sequential field
277  *
278  NXTFLD = CURFLD + 1
279  *
280  * Highlight current field data
281  *
282  XLINE = CURFLD  ;* Set the field number to display
283  ACTIVE = CURFLD ;* Set the active field number to highlight field
284  GOSUB DSPLINE
285  *
286  * Prompt for input
287  *
288  PRINT @(CONTROL(CURFLD)<1,5>,CONTROL(CURFLD)<1,6>):REVERSE:
289  INPUT IDATA(CURFLD):
290  *
```

D:\Atwin32.dev\Samples\UIEX\UIEX4

```
255 ************************************************************************
256 *
257 * The INPUT.FIELD subroutine is the main prompting routine. This routine
258 * displays the prompt string (from the CONTROL array), and enters a loop,
259 * prompting for a specified field (CURFLD) and setting the next field
260 * variable (NXTFLD) appropriately. The loop repeats until the NXTFLD
261 * (initially set to CURFLD) changes. This causes the field prompt to be
262 * repeated in case unrecognized keys are pressed (for example, the PGUP
263 * key), or invalid data is entered (illegal customer ID, etc.) If a NULL
264 * is entered for the ID field, and there is no current record, the ACTION
265 * loop control variable, DONE, and the RECORD loop control variable, XIT,
266 * are set to TRUE, and this routine exits. A mouse click is handled by
267 * setting the NXTFLD variable to the field number that was clicked.
268 *
269 INPUT.FIELD:
270 *
271 ************************************************************************
272 *
273 * Display the prompt message
274 *
275 CALL SUI.DISPLAY.LABEL(5, 22, 74, 'L', 0, CONTROL(CURFLD)<1,9>, CMD, MAT Ter
276 *
277 ************************************************************************
278 *
279 * Initialize current value, next field & character positions
280 *
281 PREVAL = IDATA(CURFLD) ;* Save previous field value to detect change in valu
282 NXTFLD = CURFLD ;* Assume field number not changed
283 BEGPOS = 0 ;* Set initial text display character position
284 CURPOS = 0 ;* Set initial text cursor character position
285 *
286 ************************************************************************
287 *
288 * Prompt for this field until NXTFLD variable is updated
289 *
290 LOOP
291  *
292  ************************************************************************
293  *
294  * Prompt for input with full text editing functions
295  *
296  CALL SUI.INPUT.TEXT(CONTROL(CURFLD)<1,5>, CONTROL(CURFLD)<1,6>, CONTROL(CUR
297  PRINT TERM.CSRON:EL: ;* Turn cursor back on & clear the error line
298  *
```

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines				
Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX3

```
291  * Check for NULL
292  *
293  K = LEN(IDATA(CURFLD))
294  IF K = 0 THEN
295   *
296   * No change if NULL entered
297   *
298   IDATA(CURFLD) = PREVAL
299  END ELSE
300   *
301   * Erase old data
302   *
303   IF K < CONTROL(CURFLD)<1,7> THEN
304    PRINT @(K+CONTROL(CURFLD)<1,5>,CONTROL(CURFLD)<1,6>):SPACE(CONTROL(CURFLD
305   END
306  END
307  *
308  * Reset display attribute
309  *
310  PRINT NORMAL:
311  *
312  * Clear the error line
313  *
314  PRINT EL:
315  *
316  ************************************************************************
317  *
318  * Check for any special values (like 'END' or 'EXIT')
319  *
320  IF OCONV(IDATA(CURFLD),'MCU') EQ 'END' THEN
321   IDATA(CURFLD) = PREVAL ;* Restore previous value
322   GOSUB CHECK.ABANDON ;* Ensure OK to loose changes
323   IF OK THEN
324    *
325    ************************************************************************
326    *
327    * No changes, or user OKs abandoning them, so we are outa here!
328    *
329    DONE = TRUE
330    XIT = TRUE
331    RETURN
332    *
333   END ELSE
334    *
335    ************************************************************************
336    *
337    * User does not want to abandon changes, so reprompt
338    *
```

D:\Atwin32.dev\Samples\UIEX\UIEX4

```
299  *******************************************************************
300  *
301  * Decode returned CMD
302  *
303  BEGIN CASE
304   *
305   ********************************************************************
306   *
307   * Check for ENTER, TAB, UP, DOWN or mouse click
308   *
309   CASE CMD EQ TERM.ENTER$ OR CMD EQ TERM.TAB$ OR CMD EQ TERM.DOWN$
310    NXTFLD = CURFLD + 1; * Move to next field
311   CASE CMD EQ TERM.STAB$ OR CMD EQ TERM.UP$
312    IF CURFLD > 1 THEN NXTFLD = CURFLD - 1 ;* Move to previous field
313   CASE CMD EQ TERM.LEFTBUTTON$
314    GOSUB CHECK.MOUSE ;* Move to clicked-on field
315    IF CURFLD EQ 1 AND NXTFLD NE CURFLD THEN
316     IF IDATA(1) EQ '' THEN NXTFLD = CURFLD ;* Mouse click in invalid field -
317    END
318   *
319  END CASE
320  *
321  ************************************************************************
322  *
323  * Check for any special values (like 'END' or 'EXIT')
324  *
325  IF OCONV(IDATA(CURFLD),'MCU') EQ 'END' THEN
326   IDATA(CURFLD) = PREVAL ;* Restore previous value
327   GOSUB CHECK.ABANDON ;* Ensure OK to loose changes
328   IF OK THEN
329    *
330    ************************************************************************
331    *
332    * No changes, or user OKs abandoning them, so we are outa here!
333    *
334    DONE = TRUE
335    XIT = TRUE
336    RETURN
337    *
338   END ELSE
339    *
340    ************************************************************************
341    *
342    * User does not want to abandon changes, so refresh previous value &
343    * reprompt
344    *
```

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines				
Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX3

```
339    NXTFLD = CURFLD ;* Reprompt
340    *
341   END
342  END
343  IF IDATA(CURFLD) EQ SPACE(LEN(IDATA(CURFLD))) THEN IDATA(CURFLD) = ''
344  *
345  ************************************************************************
346  *
347  * Peform field data validation if next field number changed
348  *
349  IF NXTFLD NE CURFLD THEN
350   BEGIN CASE
351    *
352    CASE CURFLD EQ 1
353     *
354     ***********************************************************************
355     *
356     * Validate the ID field. If NULL, quit. If changed, read new record.
357     *
358     IF CUST.ID EQ '' AND IDATA(CURFLD) EQ '' THEN
359      DONE = TRUE
360      XIT = TRUE
361      RETURN
362     END
363     *
364     IF IDATA(CURFLD) NE PREVAL THEN
365      ID = IDATA(CURFLD)
366      GOSUB CHECK.ID ;* Check the newly entered ID handling index lookups
367      IF OK THEN
368       *
369       ********************************************************************
370       *
371       * New ID (or result of index lookup) is good!
372       *
373       IDATA(CURFLD) = ID ;* Update the current field value in case of index
374       *
375      END ELSE
376       *
377       ********************************************************************
378       *
379       * New ID is invalid
380       *
381       IDATA(CURFLD) = PREVAL ;* Restore previous value

382       NXTFLD = CURFLD ;* Reprompt
383       *
384      END
385     END
386     *
387   END CASE
388  END
```

D:\Atwin32.dev\Samples\UIEX\UIEX4

```
345    XLINE = CURFLD ;* Field number to refresh
346    GOSUB DSPLINE ;* Redisplay the previous value
347    NXTFLD = CURFLD ;* Reprompt
348    *
349   END
350  END

351  *
352  ************************************************************************
353  *
354  * Peform field data validation if next field number changed
355  *
356  IF NXTFLD NE CURFLD THEN
357   BEGIN CASE
358    *
359    CASE CURFLD EQ 1
360     *
361     ***********************************************************************
362     *
363     * Validate the ID field. If NULL, quit. If changed, read new record.
364     *
365     IF CUST.ID EQ '' AND IDATA(CURFLD) EQ '' THEN
366      DONE = TRUE
367      XIT = TRUE
368      RETURN
369     END
370     *
371     IF IDATA(CURFLD) NE PREVAL THEN
372      ID = IDATA(CURFLD)
373      GOSUB CHECK.ID ;* Check the newly entered ID handling index lookups
374      IF OK THEN
375       *
376       ********************************************************************
377       *
378       * New ID (or result of index lookup) is good!
379       *
380       IDATA(CURFLD) = ID ;* Update the current field value in case of index
381       *
382      END ELSE
383       *
384       ********************************************************************
385       *
386       * New ID is invalid
387       *
388       IDATA(CURFLD) = PREVAL ;* Restore previous value
389       XLINE = CURFLD ;* Field number to refresh
390       GOSUB DSPLINE ;* Redisplay the previous value
391       NXTFLD = CURFLD ;* Reprompt
392       *
393      END
394     END
395     *
396   END CASE
397  END
```

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines				
Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX3

```
 *
WHILE CURFLD EQ NXTFLD DO REPEAT
*
* Redisplay the field data using the inactive highlighting
*
XLINE = CURFLD
ACTIVE = 0
GOSUB DSPLINE
*
RETURN
*
*
********************************************************************************
*
* The CHECK.EXIT subroutine checks if any field data has changed, and
* prompts if the user wants to abandon changes. If no changes, or the user
* decides to abandon the changes, the DONE and XIT loop control variables
* are set to TRUE, causing all three loops to terminate, and the program
* itself to exit.
*
CHECK.EXIT: *
*
GOSUB CHECK.ABANDON
IF OK THEN
 DONE = TRUE
 XIT = TRUE
 GOSUB HIDEPIX
END
RETURN
*
*
********************************************************************************
*
* The CHECK.ID subroutine validates a newly entered item-ID. If the
* current record has unsaved changes, the user is prompted to abandon the
* changes. If no changes, or the user abandons the changes, and the new ID
* is not NULL, an attempt is made to read a record using the new ID. If
* the read is not successful, the ID is assumed to be a search string, and
* the search subroutine is called to select an ID based on the search
* string. If a valid ID is returned (or if one was initially entered), the
* new record data is displayed. If the new ID is null, or if the search
* routine did not return a valid ID, a warning message is displayed and
* the OK indicator variable is set to FALSE. Otherwise it is set to TRUE.
*
CHECK.ID: *
*
********************************************************************************
*
* Make sure we don't have any unsaved data before changing the ID
*
GOSUB CHECK.ABANDON
IF NOT(OK) THEN RETURN ;* Reprompt for the ID
IF ID EQ '' THEN
 OK = FALSE ;* NULL is not a valid ID!
```

D:\Atwin32.dev\Samples\UIEX\UIEX4

```
 *
WHILE CURFLD EQ NXTFLD DO REPEAT
*
RETURN
*
*
********************************************************************************
*
* The CHECK.EXIT subroutine checks if any field data has changed, and
* prompts if the user wants to abandon changes. If no changes, or the
* user decides to abandon the changes, the DONE and XIT loop control
* variables are set to TRUE, causing all three loops to terminate, and the
* program itself to exit.
*
CHECK.EXIT: *
*
GOSUB CHECK.ABANDON
IF OK THEN
 DONE = TRUE
 XIT = TRUE
 GOSUB HIDEPIX
END
RETURN
*
*
********************************************************************************
*
* The CHECK.ID subroutine validates a newly entered item-ID. If the
* current record has unsaved changes, the user is prompted to abandon the
* changes. If no changes, or the user abandons the changes, and the new ID
* is not NULL, an attempt is made to read a record using the new ID. If
* the read is not successful, the ID is assumed to be a search string, and
* the search subroutine is called to select an ID based on the search
* string. If a valid ID is returned (or if one was initially entered), the
* new record data is displayed. If the new ID is null, or if the search
* routine did not return a valid ID, a warning message is displayed and
* the OK indicator variable is set to FALSE. Otherwise it is set to TRUE.
*
CHECK.ID: *
*
********************************************************************************
*
* Make sure we don't have any unsaved data before changing the ID
*
GOSUB CHECK.ABANDON
IF NOT(OK) THEN RETURN ;* Reprompt for the ID
IF ID EQ '' THEN
 OK = FALSE ;* NULL is not a valid ID!
```

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines				
Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX3

```
END ELSE
 *
 ************************************************************************
 *
 * Check if user wants new ID
 *
 IF ID EQ 'N' OR ID EQ 'n' THEN
  * Get next sequential ID
  READVU ID FROM FN.CUST.CTRL,'NEXT',2 THEN
   WRITEV ID + 1 ON FN.CUST.CTRL,'NEXT',2
   GOSUB RESTART
   IDATA(1) = ID
  END ELSE
   PRINT EL:UNDREV:'Next item counter record not found!':NORMAL:BEL:
   INPUT ANS:
   PRINT EL:
   OK = FALSE
   RETURN
  END
 END ELSE
  *
  ************************************************************************
  *
  * Try to read the customer record from the entered ID
  *
  GOSUB HIDEPIX ;* Hide previous cust picture before reading new one
  GOSUB READ.RECORD
  IF NOT(OK) THEN
   *
   ************************************************************************
   *
   * The attempt to read a record failed - assume the ID is a search string
   *
   CALL UIEX.GET.CUST.IDX(XID,ID,FN.CUST.XREF,FN.CUST)
   GOSUB DSPSCRN ;* Refresh the screen after index lookup
   *
   ************************************************************************
   *
   * If the user did not select an item in the search routine, reprompt
   *
   IF XID EQ '' THEN RETURN
   *
   ************************************************************************
   *
   * Try to read the customer record from the selected ID
   *
   GOSUB HIDEPIX ;* Hide previous cust picture before reading new one
   ID = XID

   GOSUB READ.RECORD
   *
  END
 END
END
```

D:\Atwin32.dev\Samples\UIEX\UIEX4

```
END ELSE
 *
 ************************************************************************
 *
 * Check if user wants new ID
 *
 IF ID EQ 'N' OR ID EQ 'n' THEN
  * Get next sequential ID
  READVU ID FROM FN.CUST.CTRL,'NEXT',2 THEN
   WRITEV ID + 1 ON FN.CUST.CTRL,'NEXT',2
   GOSUB RESTART
   IDATA(1) = ID
  END ELSE
   PRINT EL:TERM.DRV:'Next item counter record not found!':TERM.NV:BEL:
   INPUT ANS:
   PRINT EL:
   OK = FALSE
   RETURN
  END
 END ELSE
  *
  ************************************************************************
  *
  * Try to read the customer record from the entered ID
  *
  GOSUB HIDEPIX ;* Hide previous cust picture before reading new one
  GOSUB READ.RECORD
  IF NOT(OK) THEN
   *
   ************************************************************************
   *
   * The attempt to read a record failed - assume the ID is a search string
   *
   CALL UIEX.GET.CUST.IDX(XID,ID,FN.CUST.XREF,FN.CUST)
   GOSUB DSPSCRN ;* Refresh the screen after index lookup
   *
   ************************************************************************
   *
   * If the user did not select an item in the search routine, reprompt
   *
   IF XID EQ '' THEN RETURN
   *
   ************************************************************************
   *
   * Try to read the customer record from the selected ID
   *


   ID = XID
   GOSUB HIDEPIX ;* Hide previous cust picture before reading new one
   GOSUB READ.RECORD
   *
  END
 END
END
```

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines				
Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX3

```
496 *
497 ************************************************************************
498 *
499 * If success, display the new record, otherwise show warning message
500 *
501 IF OK THEN
502  GOSUB DSPDATA ;* Display new record
503 END ELSE
504  PRINT EL:UNDREV:'Please enter a valid customer ID!':NORMAL:BEL:
505 END
506 RETURN
507 *
508 *
509 ************************************************************************
510 *
511 * The READ.RECORD subroutine reads a new customer record from the file and
512 * initializes the internal field data array (IDATA) from the record array
513 * (CUST.REC). If the record does not exist, the routine returns with the
514 * OK indicator variable set to FALSE. Otherwise OK is set to TRUE, and the
515 * CUST.ID variable is set to the new ID.
516 *
517 READ.RECORD: *
518 *
519 MATREAD CUST.REC FROM FN.CUST,ID THEN
520  CUST.ID = ID
521  IDATA(1) = CUST.ID
522  IDATA(2) = CUST.CONTACT
523  IDATA(3) = CUST.NAME
524  IDATA(4) = CUST.ADDRESS1
525  IDATA(5) = CUST.ADDRESS2
526  IDATA(6) = CUST.CITY
527  IDATA(7) = CUST.ST
528  IDATA(8) = CUST.ZIP
529  IDATA(9) = CUST.COUNTRY
530  IDATA(10) = CUST.PHONE
531  IDATA(11) = CUST.FAX
532  IDATA(12) = CUST.HISTORY
533  OK = TRUE ;* Set the SUCCESS indicator
534 END ELSE
535  OK = FALSE ;* Set the FAILURE indicator
536 END
537 RETURN
538 *
539 *
540 ************************************************************************
541 *
542 * The DELETE.RECORD subroutine confirms that the user intends to delete
543 * the current record. If the action is confirmed, the CUST.DELETE
544 * subroutine is called to perform the deletion. A separate subroutine is
545 * used to handle updating indexes, etc.
546 *
547 DELETE.RECORD: *
548 *
549 IF CUST.ID NE '' THEN
```

D:\Atwin32.dev\Samples\UIEX\UIEX4

```
499 *
500 ************************************************************************
501 *
502 * If success, display the new record, otherwise show warning message
503 *
504 IF OK THEN
505  GOSUB DSPDATA ;* Display new record
506 END ELSE
507  PRINT EL:TERM.DRV:'Please enter a valid customer ID!':TERM.NV:BEL:
508 END
509 RETURN
510 *
511 *
512 ************************************************************************
513 *
514 * The READ.RECORD subroutine reads a new customer record from the file and
515 * initializes the internal field data array (IDATA) from the record array
516 * (CUST.REC). If the record does not exist, the routine returns with the
517 * OK indicator variable set to FALSE. Otherwise OK is set to TRUE, and the
518 * CUST.ID variable is set to the new ID.
519 *
520 READ.RECORD: *
521 *
522 MATREAD CUST.REC FROM FN.CUST,ID THEN
523  CUST.ID = ID
524  IDATA(1) = CUST.ID
525  IDATA(2) = CUST.CONTACT
526  IDATA(3) = CUST.NAME
527  IDATA(4) = CUST.ADDRESS1
528  IDATA(5) = CUST.ADDRESS2
529  IDATA(6) = CUST.CITY
530  IDATA(7) = CUST.ST
531  IDATA(8) = CUST.ZIP
532  IDATA(9) = CUST.COUNTRY
533  IDATA(10) = CUST.PHONE
534  IDATA(11) = CUST.FAX
535  IDATA(12) = CUST.HISTORY
536  OK = TRUE ;* Set the SUCCESS indicator
537 END ELSE
538  OK = FALSE ;* Set the FAILURE indicator
539 END
540 RETURN
541 *
542 *
543 ************************************************************************
544 *
545 * The DELETE.RECORD subroutine confirms that the user intends to delete
546 * the current record. If the action is confirmed, the CUST.DELETE
547 * subroutine is called to perform the deletion. A separate subroutine is
548 * used to handle updating indexes, etc.
549 *
550 DELETE.RECORD: *
551 *
552 IF CUST.ID NE '' THEN
```

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines				
Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX3

```
550  PRINT EL:UNDREV:'Are you sure you want to delete this customer? ':NORMAL:
551  INPUT ANS:
552  IF ANS[1,1] EQ 'Y' OR ANS[1,1] EQ 'y' THEN
553   * Deletion has been confirmed - do the delete
554   OK = TRUE ;* Set the SUCCESS indicator
555   CALL UIEX.CUST.DELETE(CUST.ID,FN.CUST,FN.CUST.XREF)
556   DONE = TRUE ;* proceed to next record
557   GOSUB HIDEPIX ;* erase picture
558  END ELSE
559   OK = FALSE ;* Set the FAILURE indicator
560  END
561 END
562 RETURN
563 *
564 *
565 ************************************************************************
566 *
567 * The SAVE.RECORD subroutine copies internal field data from the IDATA
568 * array to the customer record array (CUST.REC). The CUST.UPDATE
569 * subroutine is called to perform the update. A separate subroutine is
570 * used to handle updating indexes, etc.
571 *
572 SAVE.RECORD: *
573 *
574 IF IDATA(1) NE '' THEN
575  * Copy data from the internal field data array (IDATA) to the CUST.REC arra
576  CUST.ID = IDATA(1)
577  CUST.CONTACT = IDATA(2)
578  CUST.NAME = IDATA(3)
579  CUST.ADDRESS1 = IDATA(4)
580  CUST.ADDRESS2 = IDATA(5)
581  CUST.CITY = IDATA(6)
582  CUST.ST = IDATA(7)
583  CUST.ZIP = IDATA(8)
584  CUST.COUNTRY = IDATA(9)
585  CUST.PHONE = IDATA(10)
586  CUST.FAX = IDATA(11)
587  CUST.HISTORY = IDATA(12)
588  * Update the file
589  CALL UIEX.CUST.UPDATE(CUST.ID,MAT CUST.REC,FN.CUST,FN.CUST.XREF)
590  OK = TRUE ;* Set the SUCCESS indicator
591 END ELSE
592  OK = FALSE ;* Set the FAILURE indicator
593 END
594 DONE = TRUE ;* proceed to next record
595 RETURN
596 *
597 *
598 ************************************************************************
599 *
600 * The RESTART subroutine prepares the internal field data array (IDATA),
601 * customer record array (CUST.REC) and ID for a new customer record.
602 *
603 RESTART: *
```

D:\Atwin32.dev\Samples\UIEX\UIEX4

```
553  PRINT EL:TERM.DRV:'Are you sure you want to delete this customer? ':TERM.NV
554  INPUT ANS:
555  IF ANS[1,1] EQ 'Y' OR ANS[1,1] EQ 'y' THEN
556   * Deletion has been confirmed - do the delete
557   OK = TRUE ;* Set the SUCCESS indicator
558   CALL UIEX.CUST.DELETE(CUST.ID,FN.CUST,FN.CUST.XREF)
559   DONE = TRUE ;* proceed to next record
560   GOSUB HIDEPIX ;* erase picture
561  END ELSE
562   OK = FALSE ;* Set the FAILURE indicator
563  END
564 END
565 RETURN
566 *
567 *
568 ************************************************************************
569 *
570 * The SAVE.RECORD subroutine copies internal field data from the IDATA
571 * array to the customer record array (CUST.REC). The CUST.UPDATE
572 * subroutine is called to perform the update. A separate subroutine is
573 * used to handle updating indexes, etc.
574 *
575 SAVE.RECORD: *
576 *
577 IF IDATA(1) NE '' THEN
578  * Copy data from the internal field data array (IDATA) to the CUST.REC arra
579  CUST.ID = IDATA(1)
580  CUST.CONTACT = IDATA(2)
581  CUST.NAME = IDATA(3)
582  CUST.ADDRESS1 = IDATA(4)
583  CUST.ADDRESS2 = IDATA(5)
584  CUST.CITY = IDATA(6)
585  CUST.ST = IDATA(7)
586  CUST.ZIP = IDATA(8)
587  CUST.COUNTRY = IDATA(9)
588  CUST.PHONE = IDATA(10)
589  CUST.FAX = IDATA(11)
590  CUST.HISTORY = IDATA(12)
591  * Update the file
592  CALL UIEX.CUST.UPDATE(CUST.ID,MAT CUST.REC,FN.CUST,FN.CUST.XREF)
593  OK = TRUE ;* Set the SUCCESS indicator
594 END ELSE
595  OK = FALSE ;* Set the FAILURE indicator
596 END
597 DONE = TRUE ;* proceed to next record
598 RETURN
599 *
600 *
601 ************************************************************************
602 *
603 * The RESTART subroutine prepares the internal field data array (IDATA),
604 * customer record array (CUST.REC) and ID for a new customer record.
605 *
606 RESTART: *
```

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines				
Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX3

```
*
CUST.ID = ''
MAT CUST.REC = ''
MAT IDATA = ''
ACTIVE = 0
RETURN
*
*
*************************************************************************
*
* The CHECK.CHANGED subroutine checks if any internal field data has been
* changed. The OK indicator variable is set to FALSE if any data is
* changed, otherwise it is set to TRUE. The ID field is not checked, since
* it is appropriately handled by the CHECK.ID subroutine.
*
CHECK.CHANGED: *
*
OK = FALSE
IF CUST.CONTACT # IDATA(2) THEN RETURN
IF CUST.NAME # IDATA(3) THEN RETURN
IF CUST.ADDRESS1 # IDATA(4) THEN RETURN
IF CUST.ADDRESS2 # IDATA(5) THEN RETURN
IF CUST.CITY # IDATA(6) THEN RETURN
IF CUST.ST # IDATA(7) THEN RETURN
IF CUST.ZIP # IDATA(8) THEN RETURN
IF CUST.COUNTRY # IDATA(9) THEN RETURN
IF CUST.PHONE # IDATA(10) THEN RETURN
IF CUST.FAX # IDATA(11) THEN RETURN
IF CUST.HISTORY # IDATA(12) THEN RETURN
OK = TRUE
RETURN
*
*
*************************************************************************
*
* The CHECK.ABANDON subroutine calls CHECK.CHANGED. If any changes are
* found, a message is displayed and the user is prompted to abandon the
* changes. If there are no changes, or if the user decides to abandon
* changes, the OK indicator variable is set to TRUE. Otherwise it is set
* to FALSE.
*
CHECK.ABANDON: *
*
GOSUB CHECK.CHANGED
IF NOT(OK) THEN
 PRINT EL:UNDREV:'Do you want to abandon all your changes? ':NORMAL:BEL:
 INPUT ANS:
 IF ANS[1,1] EQ 'Y' OR ANS[1,1] EQ 'y' THEN OK = 1
 PRINT EL:
END
RETURN
*
*
*************************************************************************
```

D:\Atwin32.dev\Samples\UIEX\UIEX4

```
*
CUST.ID = ''
MAT CUST.REC = ''
MAT IDATA = ''

RETURN
*
*
*************************************************************************
*
* The CHECK.CHANGED subroutine checks if any internal field data has been
* changed. The OK indicator variable is set to FALSE if any data is
* changed, otherwise it is set to TRUE. The ID field is not checked, since
* it is appropriately handled by the CHECK.ID subroutine.
*
CHECK.CHANGED: *
*
OK = FALSE
IF CUST.CONTACT # IDATA(2) THEN RETURN
IF CUST.NAME # IDATA(3) THEN RETURN
IF CUST.ADDRESS1 # IDATA(4) THEN RETURN
IF CUST.ADDRESS2 # IDATA(5) THEN RETURN
IF CUST.CITY # IDATA(6) THEN RETURN
IF CUST.ST # IDATA(7) THEN RETURN
IF CUST.ZIP # IDATA(8) THEN RETURN
IF CUST.COUNTRY # IDATA(9) THEN RETURN
IF CUST.PHONE # IDATA(10) THEN RETURN
IF CUST.FAX # IDATA(11) THEN RETURN
IF CUST.HISTORY # IDATA(12) THEN RETURN
OK = TRUE
RETURN
*
*
*************************************************************************
*
* The CHECK.ABANDON subroutine calls CHECK.CHANGED. If any changes are
* found, a message is displayed and the user is prompted to abandon the
* changes. If there are no changes, or if the user decides to abandon
* changes, the OK indicator variable is set to TRUE. Otherwise it is set
* to FALSE.
*
CHECK.ABANDON: *
*
GOSUB CHECK.CHANGED
IF NOT(OK) THEN
 PRINT EL:TERM.DRV:'Do you want to abandon all your changes? ':TERM.NV:BEL:
 INPUT ANS:
 IF ANS[1,1] EQ 'Y' OR ANS[1,1] EQ 'y' THEN OK = 1
 PRINT EL:
END
RETURN
*
*
*************************************************************************
```

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines				
Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX3

```
*
* The DSPSCRN subroutine is used to refresh the entire screen.
*
DSPSCRN: *
*
* Clear the screen
PRINT CLR:NORMAL:
*
* If running AccuTerm, display the logo
IF IMGFLG THEN
 PRINT ESC:STX:'iL,':PIX.PATH:'ASE-LOGO-SMALL.JPG,65,0,12,2,0,N':CR:
END
*
* Display heading & labels
PRINT @(5,0):'Customer File Maintenance':
FOR XLINE = 1 TO NUMFLDS
 LBWD = CONTROL(XLINE)<1,4>
 LBTX = (XLINE 'L#2 ') : CONTROL(XLINE)<1,1> ;* Prepend line number to label
 IF LBWD > 0 THEN LBTX = (LBTX : STR('.',LBWD))[1,LBWD] ;* Pad label with dots
 PRINT @(CONTROL(XLINE)<1,2>,CONTROL(XLINE)<1,3>):LBTX:
NEXT XLINE
*
* Display the field data
GOSUB DSPDATA
RETURN
*
*
*************************************************************************
*
* The DSPDATA subroutine is used to refresh the field data for all fields.
*
DSPDATA: *
*
FOR XLINE = 1 TO NUMFLDS
 GOSUB DSPLINE
NEXT XLINE
GOSUB SHOWPIX
RETURN
*
*
*************************************************************************
*
* The DSPLINE subroutine is used to refresh the field data for one field.
* The field to be refreshed is specified by the XLINE variable. If the
* field is active (XLINE = ACTIVE), then the field data is displayed using
* the REVERSE display attribute. Otherwise it is displayed using the
* DIMREV display attribute.
*
DSPLINE: *
*
MSK = 'L#':CONTROL(XLINE)<1,7>
PRINT @(CONTROL(XLINE)<1,5>,CONTROL(XLINE)<1,6>):
IF XLINE EQ ACTIVE THEN
 PRINT REVERSE:
```

D:\Atwin32.dev\Samples\UIEX\UIEX4

```
*
* The DSPSCRN subroutine is used to refresh the entire screen.
*
DSPSCRN: *
*
* Clear the screen
PRINT TERM.CLEAR:
*
* If running AccuTerm, display the logo
IF IMGFLG THEN
 PRINT ESC:STX:'iL,':PIX.PATH:'ASE-LOGO-SMALL.JPG,65,0,12,2,0,N':CR:
END
*
* Display heading & labels
CALL SUI.DISPLAY.LABEL(5, 0, 0, 'L', 0, 'Customer File Maintenance', CMD, MA
FOR XLINE = 1 TO NUMFLDS
 LBWD = CONTROL(XLINE)<1,4>
 LBTX = (XLINE 'L#2 ') : CONTROL(XLINE)<1,1> ;* Prepend line number to label
 IF LBWD > 0 THEN LBTX = (LBTX : STR('.',LBWD))[1,LBWD] ;* Pad label with dots
 CALL SUI.DISPLAY.LABEL(CONTROL(XLINE)<1,2>, CONTROL(XLINE)<1,3>, LBWD, 'L',
NEXT XLINE
*
* Display the field data
GOSUB DSPDATA
RETURN
*
*
*************************************************************************
*
* The DSPDATA subroutine is used to refresh the field data for all fields.
*
DSPDATA: *
*
FOR XLINE = 1 TO NUMFLDS
 GOSUB DSPLINE
NEXT XLINE
GOSUB SHOWPIX
RETURN
*
*
*************************************************************************
*
* The DSPLINE subroutine is used to refresh the field data for one field.
* The field to be refreshed is specified by the XLINE variable.
*
DSPLINE: *
*
CALL SUI.DISPLAY.TEXT(CONTROL(XLINE)<1,5>, CONTROL(XLINE)<1,6>, CONTROL(XLIN
```

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines

Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX3

```
712 END ELSE
713  PRINT DIMREV:
714 END
715 PRINT IDATA(XLINE) MSK:
716 PRINT NORMAL:
717 RETURN
718 *
719 *
720 ************************************************************************
721 *
722 * If running AccuTerm and an customer ID has been entered, try to display
723 * the customer's picture for the current record.
724 *
725 SHOWPIX: *
726 *
727 IF IMGFLG THEN
728  IF CUST.ID NE '' THEN
729   PRINT ESC:STX:'iL,':PIX.PATH:'CUST\':CUST.ID:'.JPG,60,3,15,8,1,B':CR:
730  END
731 END
732 RETURN
733 *
734 *
735 ************************************************************************
736 *
737 * Erase the current customer picture before calling the lookup screen
738 * because pictures always display over text.
739 *
740 HIDEPIX: *
741 *
742 IF IMGFLG THEN
743  IF CUST.ID NE '' THEN
744   PRINT ESC:STX:'iD,':PIX.PATH:'CUST\':CUST.ID:'.JPG':CR:
745  END
746 END
```

D:\Atwin32.dev\Samples\UIEX\UIEX4

```
708 RETURN
709 *
710 *
711 ************************************************************************
712 *
713 * If running AccuTerm and an customer ID has been entered, try to display
714 * the customer's picture for the current record.
715 *
716 SHOWPIX: *
717 *
718 IF IMGFLG THEN
719  IF CUST.ID NE '' THEN
720   PRINT ESC:STX:'iL,':PIX.PATH:'CUST\':CUST.ID:'.JPG,60,3,15,8,1,B':CR:
721  END
722 END
723 RETURN
724 *
725 *
726 ************************************************************************
727 *
728 * Erase the current customer picture before calling the lookup screen
729 * because pictures always display over text.
730 *
731 HIDEPIX: *
732 *
733 IF IMGFLG THEN
734  IF CUST.ID NE '' THEN
735   PRINT ESC:STX:'iD,':PIX.PATH:'CUST\':CUST.ID:'.JPG':CR:
736  END
737 END
738 RETURN
739 *
740 *
741 ************************************************************************
742 *
743 * The CHECK.MOUSE subroutine is called if any SUI input routine returns
744 * with CMD set to TERM.LEFTBUTTON$. This routine checks each field,
745 * passing the field position and size to SUI.MOUSE.HIT. If a "hit" is
746 * detected, the NXTFLD variable is set to the field number of the clicked
747 * field. If no field was clicked, the position and size of the command bar
748 * is checked, and if "hit", NXTFLG is set to NUMFLDS + 1 to cause the
749 * ACTION and PROMPT loops to prompt for action using the command bar. The
750 * actual mouse click location is saved internally by the SUI input
751 * routines, and if a command button is clicked while a different field is
752 * active, the click action is executed when SUI.INPUT.BTNBAR is called.
753 *
754 CHECK.MOUSE: *
755 *
756 FOR XLINE = 1 TO NUMFLDS
```

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines				
Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX3		D:\Atwin32.dev\Samples\UIEX\UIEX4	
		757	CALL SUI.MOUSE.HIT(CONTROL(XLINE)<1,5>, CONTROL(XLINE)<1,6>, CONTROL(XLINE)
		758	IF HIT THEN
		759	NXTFLD = XLINE
		760	RETURN
		761	END
		762	NEXT XLINE
747	RETURN	763	RETURN
748	*	764	*
749	END	765	END
750		766	

Found 44 differences (1 moved block): 177 lines, 263 inline differences in 95 changed lines				
Added(61,70)	Deleted(21,64)	Changed(95)	Changed in changed(129)	Ignored