

```

D:\Atwin32.dev\Samples\UIEX\UIEX2
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3 * Copyright (c) 2004 Zumasys, Inc. as an unpublished work.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 2
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.
32 *
33 *****
34 *
35 * The CUST.REC INCLUDE item declares the CUST.REC array, which is used to
36 * store a working copy of the current data record. It also defines the
37 * record layout by equating field names to array positions in the CUST.REC
38 * array.
39 *
40 $INCLUDE UIEX.CUST.REC
41 *
42 *****
43 *
44 * EQUates for control characters and delimiters and other constants
45 *
46 EQU VM TO CHAR(253)
47 EQU BEL TO CHAR(7)
48 EQU ESC TO CHAR(27)
49 EQU FALSE TO 0
50 EQU TRUE TO 1
51 *
52 *****
53 *
54 * Open our files...

```

```

D:\Atwin32.dev\Samples\UIEX\UIEX2
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3 * Copyright (c) 2004 Zumasys, Inc. as an unpublished work.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 2
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.
32 *
33 *****
34 *
35 * The CUST.REC INCLUDE item declares the CUST.REC array, which is used to
36 * store a working copy of the current data record. It also defines the
37 * record layout by equating field names to array positions in the CUST.REC
38 * array.
39 *
40 $INCLUDE UIEX.CUST.REC
41 *
42 *****
43 *
44 * EQUates for control characters and delimiters and other constants
45 *
46 EQU VM TO CHAR(253)
47 EQU BEL TO CHAR(7)
48 EQU ESC TO CHAR(27)
49 EQU FALSE TO 0
50 EQU TRUE TO 1
51 *
52 *****
53 *
54 * Open our files...

```

No differences found

Added(0,0)

Deleted(0,0)

Changed(0)

Changed in changed(0)

Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

55 *
56 OPEN 'CUST.SAMPLE' TO FN.CUST ELSE PRINT 'NO CUST.SAMPLE FILE'; STOP
57 OPEN 'CUST.SAMPLE.CTRL' TO FN.CUST.CTRL ELSE PRINT 'NO CUST.SAMPLE.CTRL FILE'
58 OPEN 'CUST.SAMPLE.XREF' TO FN.CUST.XREF ELSE PRINT 'NO CUST.SAMPLE.XREF'; ST
59 *
60 *****
61 *
62 * Define the data field control structures. NUMFLDS is the number of
63 * fields. The CONTROL array defines the field control elements such as
64 * field label, label position, field position and size, and field prompt.
65 * The IDATA array stores the current field values, and is initialized from
66 * the CUST.REC array when a data record is read from the file. When the
67 * record is updated, values are copied from the IDATA array to the
68 * CUST.REC array and the CUST.REC array is passed to the CUST.UPDATE
69 * subroutine to update the data and index files (a separate subroutine is
70 * used to update the file since the update process may handle other
71 * functions like updating indexes).
72 *
73 NUMFLDS = 12
74 DIM CONTROL(12)
75 DIM IDATA(12)
76 *
77 * Define field control elements
78 * value 1: field label text
79 * value 2: field label column
80 * value 3: field label row
81 * value 4: field label width (0 for actual width, no padding)
82 * value 5: field data column
83 * value 6: field data row
84 * value 7: field data width
85 * value 8: field data height
86 * value 9: field prompt message
87 CONTROL(1) = 'Customer ID???0?8??0?ýEnter the ID, or name to search for, or
88 CONTROL(2) = 'Contact???0?8??0?ýEnter the contact name'
89 CONTROL(3) = 'Company name???0?8??0?ýEnter the company name'
90 CONTROL(4) = 'Address line 1???0?8??0?ýEnter address line 1'
91 CONTROL(5) = 'Address line 2???0?8??0?ýEnter address line 2'
92 CONTROL(6) = 'City???0?8??5?ýEnter the city'
93 CONTROL(7) = 'State or province???0?8??5?ýEnter the state or province abbrev
94 CONTROL(8) = 'Zip/postal code???0?8?0?0?ýEnter the zip or postal code'
95 CONTROL(9) = 'Country??1?0?8?1?8?ýEnter the country'
96 CONTROL(10) = 'Phone??2?0?8?2?5?ýEnter the phone number'
97 CONTROL(11) = 'Fax??3?0?8?3?5?ýEnter the fax number'
98 CONTROL(12) = 'Notes??4?0?8?4?0?ýEnter any notes about this customer'
99 *
100 *****
101 *
102 * Define some screen control strings for prompts & errors
103 *
104 PROMPT ''
105 CLR = @(-1) ;* Clear entire screen
106 CEOL = @(-4) ;* Clear to end of line
107 PL = @(5,22):CEOL ;* Prompt line
108 EL = @(5,23):CEOL ;* Error line

```

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

55 *
56 OPEN 'CUST.SAMPLE' TO FN.CUST ELSE PRINT 'NO CUST.SAMPLE FILE'; STOP
57 OPEN 'CUST.SAMPLE.CTRL' TO FN.CUST.CTRL ELSE PRINT 'NO CUST.SAMPLE.CTRL FILE'
58 OPEN 'CUST.SAMPLE.XREF' TO FN.CUST.XREF ELSE PRINT 'NO CUST.SAMPLE.XREF'; ST
59 *
60 *****
61 *
62 * Define the data field control structures. NUMFLDS is the number of
63 * fields. The CONTROL array defines the field control elements such as
64 * field label, label position, field position and size, and field prompt.
65 * The IDATA array stores the current field values, and is initialized from
66 * the CUST.REC array when a data record is read from the file. When the
67 * record is updated, values are copied from the IDATA array to the
68 * CUST.REC array and the CUST.REC array is passed to the CUST.UPDATE
69 * subroutine to update the data and index files (a separate subroutine is
70 * used to update the file since the update process may handle other
71 * functions like updating indexes).
72 *
73 NUMFLDS = 12
74 DIM CONTROL(12)
75 DIM IDATA(12)
76 *
77 * Define field control elements
78 * value 1: field label text
79 * value 2: field label column
80 * value 3: field label row
81 * value 4: field label width (0 for actual width, no padding)
82 * value 5: field data column
83 * value 6: field data row
84 * value 7: field data width
85 * value 8: field data height
86 * value 9: field prompt message
87 CONTROL(1) = 'Customer ID???0?8??0?ýEnter the ID, or name to search for, or
88 CONTROL(2) = 'Contact???0?8??0?ýEnter the contact name'
89 CONTROL(3) = 'Company name???0?8??0?ýEnter the company name'
90 CONTROL(4) = 'Address line 1???0?8??0?ýEnter address line 1'
91 CONTROL(5) = 'Address line 2???0?8??0?ýEnter address line 2'
92 CONTROL(6) = 'City???0?8??5?ýEnter the city'
93 CONTROL(7) = 'State or province???0?8??5?ýEnter the state or province abbrev
94 CONTROL(8) = 'Zip/postal code???0?8?0?0?ýEnter the zip or postal code'
95 CONTROL(9) = 'Country??1?0?8?1?8?ýEnter the country'
96 CONTROL(10) = 'Phone??2?0?8?2?5?ýEnter the phone number'
97 CONTROL(11) = 'Fax??3?0?8?3?5?ýEnter the fax number'
98 CONTROL(12) = 'Notes??4?0?8?4?0?ýEnter any notes about this customer'
99 *
100 *****
101 *
102 * Define some screen control strings for prompts & errors
103 *
104 PROMPT ''
105 CLR = @(-1) ;* Clear entire screen
106 CEOL = @(-4) ;* Clear to end of line
107 PL = @(5,22):CEOL ;* Prompt line
108 EL = @(5,23):CEOL ;* Error line

```

No differences found

Added(0,0)

Deleted(0,0)

Changed(0)

Changed in changed(0)

Ignored

```

D:\Atwin32.dev\Samples\UIEX\UIEX2
109 *
110 * Wyse 60 attributes for NORMAL, REVERSE, DIM REVERSE and UNDERLINE
111 * REVERSE (we like wyse 60 because the attributes dont take any space on
112 * the screen, and more than one attribute can be displayed at one time).
113 * If running AccuTerm in Viewpoint A2 Enhanced emulation, you can use the
114 * ADDS 4000 attributes instead. Just change the upper-case "G" below to
115 * lower-case "g".
116 *
117 NORMAL = ESC:'G0' ;* Normal - display headings & field labels
118 REVERSE = ESC:'G4' ;* Reverse - display active field data
119 DIMREV = ESC:'Gt' ;* Dim Reverse - display inactive field data
120 UNDREV = ESC:'G<' ;* Underline Reverse - display warnings
121 *
122 *****
123 *
124 * This program processes one customer record at a time, and is organized
125 * using three nested loops. The outermost loop (the RECORD loop) is
126 * executed once for each record accessed. The loop repeats until the XIT
127 * control variable is set to TRUE.
128 *
129 XIT = FALSE
130 LOOP UNTIL XIT DO
131 *
132 *****
133 *
134 * Before prompting for the customer ID, reset the internal data array
135 * (IDATA) and CUST.ID variables, then clear the screen and display the
136 * heading and field labels.
137 *
138 GOSUB RESTART
139 GOSUB DSPSCRN
140 *
141 *****
142 *
143 * The middle loop is the ACTION loop. It is executed for the current
144 * record until the user performs an action that terminates processing of
145 * that record, such as exiting, cancelling, saving or deleting the
146 * record. The field number to begin prompting (NXTFLD) is initialized to
147 * 1, causing the ID field to be prompted first. The loop immediately
148 * enters the PROMPT loop, followed by the field modification prompt. The
149 * loop repeats until the DONE control variable is set to TRUE.
150 *
151 NXTFLD = 1
152 DONE = FALSE
153 LOOP
154 *
155 *****
156 *
157 * The inner loop is the PROMPT loop. It is executed for each prompt
158 * field as specified by the NXTFLD variable. The prompt loop simply
159 * calls the local INPUT.FIELD subroutine which performs field input,
160 * data validation and keyboard command decoding. The FIELD loop repeats
161 * until the field number is greater than the number of fields, or until
162 * the DONE control variable is set to TRUE. The DONE control variable

```

```

D:\Atwin32.dev\Samples\UIEX\UIEX2
109 *
110 * Wyse 60 attributes for NORMAL, REVERSE, DIM REVERSE and UNDERLINE
111 * REVERSE (we like wyse 60 because the attributes dont take any space on
112 * the screen, and more than one attribute can be displayed at one time).
113 * If running AccuTerm in Viewpoint A2 Enhanced emulation, you can use the
114 * ADDS 4000 attributes instead. Just change the upper-case "G" below to
115 * lower-case "g".
116 *
117 NORMAL = ESC:'G0' ;* Normal - display headings & field labels
118 REVERSE = ESC:'G4' ;* Reverse - display active field data
119 DIMREV = ESC:'Gt' ;* Dim Reverse - display inactive field data
120 UNDREV = ESC:'G<' ;* Underline Reverse - display warnings
121 *
122 *****
123 *
124 * This program processes one customer record at a time, and is organized
125 * using three nested loops. The outermost loop (the RECORD loop) is
126 * executed once for each record accessed. The loop repeats until the XIT
127 * control variable is set to TRUE.
128 *
129 XIT = FALSE
130 LOOP UNTIL XIT DO
131 *
132 *****
133 *
134 * Before prompting for the customer ID, reset the internal data array
135 * (IDATA) and CUST.ID variables, then clear the screen and display the
136 * heading and field labels.
137 *
138 GOSUB RESTART
139 GOSUB DSPSCRN
140 *
141 *****
142 *
143 * The middle loop is the ACTION loop. It is executed for the current
144 * record until the user performs an action that terminates processing of
145 * that record, such as exiting, cancelling, saving or deleting the
146 * record. The field number to begin prompting (NXTFLD) is initialized to
147 * 1, causing the ID field to be prompted first. The loop immediately
148 * enters the PROMPT loop, followed by the field modification prompt. The
149 * loop repeats until the DONE control variable is set to TRUE.
150 *
151 NXTFLD = 1
152 DONE = FALSE
153 LOOP
154 *
155 *****
156 *
157 * The inner loop is the PROMPT loop. It is executed for each prompt
158 * field as specified by the NXTFLD variable. The prompt loop simply
159 * calls the local INPUT.FIELD subroutine which performs field input,
160 * data validation and keyboard command decoding. The FIELD loop repeats
161 * until the field number is greater than the number of fields, or until
162 * the DONE control variable is set to TRUE. The DONE control variable

```

No differences found

Added(0,0)

Deleted(0,0)

Changed(0)

Changed in changed(0)

Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

163 * may be set to TRUE in the INPUT.FIELD subroutine, if the user enters a
164 * NULL to for the item-ID.
165 *
166 LOOP
167   CURFLD = NXTFLD
168 UNTIL DONE OR CURFLD > NUMFLDS DO
169   *
170   *****
171   *
172   * Prompt for the next field. Update the NXTFLD variable with the field
173   * number to prompt next.
174   *
175   GOSUB INPUT.FIELD
176 REPEAT ;* end of PROMPT loop
177 *
178 *****
179 *
180 * If the FIELD loop exited with the DONE control variable set to TRUE,
181 * bypass the modification prompt because no action is required.
182 * Otherwise, prompt for which field to modify, or other actions such as
183 * save or delete.
184 *
185 IF NOT(DONE) THEN
186 *
187   NXTFLD = NUMFLDS + 1 ;* assume we need to reprompt for action
188 *
189   PRINT PL:'Enter field number to modify or FI to save, DE to delete, EX to
190   INPUT ANS:
191   PRINT EL:
192   *
193   *****
194   *
195   * Decode the response
196   *
197   ANS = OCONV(ANS, 'MCU')
198   BEGIN CASE
199     CASE NUM(ANS) AND ANS >= 1 AND ANS <= NUMFLDS; NXTFLD = ANS
200     CASE ANS EQ 'FI'; GOSUB SAVE.RECORD
201     CASE ANS EQ 'DE'; GOSUB DELETE.RECORD
202     CASE ANS EQ 'EX' OR ANS EQ ''; GOSUB CHECK.EXIT
203   END CASE
204   *
205 END
206 *
207 UNTIL DONE DO REPEAT ;* end of ACTION loop
208 *
209 REPEAT ;* end of RECORD loop
210 *
211 *****
212 *
213 * All done - clear the screen and exit!
214 PRINT CLR:
215 STOP
216 *

```

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

163 * may be set to TRUE in the INPUT.FIELD subroutine, if the user enters a
164 * NULL to for the item-ID.
165 *
166 LOOP
167   CURFLD = NXTFLD
168 UNTIL DONE OR CURFLD > NUMFLDS DO
169   *
170   *****
171   *
172   * Prompt for the next field. Update the NXTFLD variable with the field
173   * number to prompt next.
174   *
175   GOSUB INPUT.FIELD
176 REPEAT ;* end of PROMPT loop
177 *
178 *****
179 *
180 * If the FIELD loop exited with the DONE control variable set to TRUE,
181 * bypass the modification prompt because no action is required.
182 * Otherwise, prompt for which field to modify, or other actions such as
183 * save or delete.
184 *
185 IF NOT(DONE) THEN
186 *
187   NXTFLD = NUMFLDS + 1 ;* assume we need to reprompt for action
188 *
189   PRINT PL:'Enter field number to modify or FI to save, DE to delete, EX to
190   INPUT ANS:
191   PRINT EL:
192   *
193   *****
194   *
195   * Decode the response
196   *
197   ANS = OCONV(ANS, 'MCU')
198   BEGIN CASE
199     CASE NUM(ANS) AND ANS >= 1 AND ANS <= NUMFLDS; NXTFLD = ANS
200     CASE ANS EQ 'FI'; GOSUB SAVE.RECORD
201     CASE ANS EQ 'DE'; GOSUB DELETE.RECORD
202     CASE ANS EQ 'EX' OR ANS EQ ''; GOSUB CHECK.EXIT
203   END CASE
204   *
205 END
206 *
207 UNTIL DONE DO REPEAT ;* end of ACTION loop
208 *
209 REPEAT ;* end of RECORD loop
210 *
211 *****
212 *
213 * All done - clear the screen and exit!
214 PRINT CLR:
215 STOP
216 *

```

No differences found

Added(0,0)

Deleted(0,0)

changed(0)

changed in changed(0)

Ignored

```

D:\Atwin32.dev\Samples\UIEX\UIEX2
217 *
218 *****
219 *****
220 * LOCAL SUBROUTINES
221 *****
222 *****
223 *
224 *
225 *****
226 *
227 * The INPUT.FIELD subroutine is the main prompting routine. This routine
228 * displays the prompt string (from the CONTROL array), and enters a loop,
229 * prompting for a specified field (CURFLD) and setting the next field
230 * variable (NXTFLD) appropriately. The loop repeats until the NXTFLD
231 * (initially set to CURFLD) changes. This causes the field prompt to be
232 * repeated in case invalid data is entered (illegal customer ID, etc.) If
233 * a NULL is entered for the ID field, and there is no current record, the
234 * ACTION loop control variable, DONE, and the RECORD loop control
235 * variable, XIT, are set to TRUE, and this routine exits.
236 *
237 INPUT.FIELD:
238 *
239 *****
240 *
241 * Display the prompt message
242 *
243 PRINT PL:CONTROL(CURFLD)<1,9>:
244 *
245 *****
246 *
247 * Initialize current value, next field
248 *
249 PREVAL = IDATA(CURFLD) ;* Save previous field value to detect change in val
250 *
251 *****
252 *
253 * Prompt for this field until NXTFLD variable is updated
254 *
255 LOOP
256 *
257 *****
258 *
259 * Assume next field number is next sequential field
260 *
261 NXTFLD = CURFLD + 1
262 *
263 * Highlight current field data
264 *
265 XLINE = CURFLD ;* Set the field number to display
266 ACTIVE = CURFLD ;* Set the active field number to highlight field
267 GOSUB DSPLINE
268 *
269 * Prompt for input
270 *

```

```

D:\Atwin32.dev\Samples\UIEX\UIEX2
217 *
218 *****
219 *****
220 * LOCAL SUBROUTINES
221 *****
222 *****
223 *
224 *
225 *****
226 *
227 * The INPUT.FIELD subroutine is the main prompting routine. This routine
228 * displays the prompt string (from the CONTROL array), and enters a loop,
229 * prompting for a specified field (CURFLD) and setting the next field
230 * variable (NXTFLD) appropriately. The loop repeats until the NXTFLD
231 * (initially set to CURFLD) changes. This causes the field prompt to be
232 * repeated in case invalid data is entered (illegal customer ID, etc.) If
233 * a NULL is entered for the ID field, and there is no current record, the
234 * ACTION loop control variable, DONE, and the RECORD loop control
235 * variable, XIT, are set to TRUE, and this routine exits.
236 *
237 INPUT.FIELD:
238 *
239 *****
240 *
241 * Display the prompt message
242 *
243 PRINT PL:CONTROL(CURFLD)<1,9>:
244 *
245 *****
246 *
247 * Initialize current value, next field
248 *
249 PREVAL = IDATA(CURFLD) ;* Save previous field value to detect change in val
250 *
251 *****
252 *
253 * Prompt for this field until NXTFLD variable is updated
254 *
255 LOOP
256 *
257 *****
258 *
259 * Assume next field number is next sequential field
260 *
261 NXTFLD = CURFLD + 1
262 *
263 * Highlight current field data
264 *
265 XLINE = CURFLD ;* Set the field number to display
266 ACTIVE = CURFLD ;* Set the active field number to highlight field
267 GOSUB DSPLINE
268 *
269 * Prompt for input
270 *

```

No differences found

Added(0,0)

Deleted(0,0)

Changed(0)

Changed in changed(0)

Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

271 PRINT @(CONTROL(CURFLD)<1,5>,CONTROL(CURFLD)<1,6>):REVERSE:
272 INPUT IDATA(CURFLD):
273 *
274 * Check for NULL
275 *
276 K = LEN(IDATA(CURFLD))
277 IF K = 0 THEN
278 *
279 * No change if NULL entered
280 *
281 IDATA(CURFLD) = PREVAL
282 END ELSE
283 *
284 * Erase old data
285 *
286 IF K < CONTROL(CURFLD)<1,7> THEN
287 PRINT @(K+CONTROL(CURFLD)<1,5>,CONTROL(CURFLD)<1,6>):SPACE(CONTROL(CURFLD)
288 END
289 END
290 *
291 * Reset display attribute
292 *
293 PRINT NORMAL:
294 *
295 * Clear the error line
296 *
297 PRINT EL:
298 *
299 *****
300 *
301 * Check for any special values (like 'END' or 'EXIT')
302 *
303 IF OCONV(IDATA(CURFLD),'MCU') EQ 'END' THEN
304 IDATA(CURFLD) = PREVAL ;* Restore previous value
305 GOSUB CHECK.ABANDON ;* Ensure OK to loose changes
306 IF OK THEN
307 *
308 *****
309 *
310 * No changes, or user OKs abandoning them, so we are outa here!
311 *
312 DONE = TRUE
313 XIT = TRUE
314 RETURN
315 *
316 END ELSE
317 *
318 *****
319 *
320 * User does not want to abandon changes, so reprompt
321 *
322 NXTFLD = CURFLD ;* Reprompt
323 *
324 END

```

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

271 PRINT @(CONTROL(CURFLD)<1,5>,CONTROL(CURFLD)<1,6>):REVERSE:
272 INPUT IDATA(CURFLD):
273 *
274 * Check for NULL
275 *
276 K = LEN(IDATA(CURFLD))
277 IF K = 0 THEN
278 *
279 * No change if NULL entered
280 *
281 IDATA(CURFLD) = PREVAL
282 END ELSE
283 *
284 * Erase old data
285 *
286 IF K < CONTROL(CURFLD)<1,7> THEN
287 PRINT @(K+CONTROL(CURFLD)<1,5>,CONTROL(CURFLD)<1,6>):SPACE(CONTROL(CURFLD)
288 END
289 END
290 *
291 * Reset display attribute
292 *
293 PRINT NORMAL:
294 *
295 * Clear the error line
296 *
297 PRINT EL:
298 *
299 *****
300 *
301 * Check for any special values (like 'END' or 'EXIT')
302 *
303 IF OCONV(IDATA(CURFLD),'MCU') EQ 'END' THEN
304 IDATA(CURFLD) = PREVAL ;* Restore previous value
305 GOSUB CHECK.ABANDON ;* Ensure OK to loose changes
306 IF OK THEN
307 *
308 *****
309 *
310 * No changes, or user OKs abandoning them, so we are outa here!
311 *
312 DONE = TRUE
313 XIT = TRUE
314 RETURN
315 *
316 END ELSE
317 *
318 *****
319 *
320 * User does not want to abandon changes, so reprompt
321 *
322 NXTFLD = CURFLD ;* Reprompt
323 *
324 END

```

No differences found

Added(0,0)

Deleted(0,0)

changed(0)

changed in changed(0)

Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

325 END
326 IF IDATA(CURFLD) EQ SPACE(LEN(IDATA(CURFLD))) THEN IDATA(CURFLD) = ''
327 *
328 *****
329 *
330 * Perform field data validation if next field number changed
331 *
332 IF NXTFLD NE CURFLD THEN
333 BEGIN CASE
334 *
335 CASE CURFLD EQ 1
336 *
337 *****
338 *
339 * Validate the ID field. If NULL, quit. If changed, read new record.
340 *
341 IF CUST.ID EQ '' AND IDATA(CURFLD) EQ '' THEN
342 DONE = TRUE
343 XIT = TRUE
344 RETURN
345 END
346 *
347 IF IDATA(CURFLD) NE PREVAL THEN
348 ID = IDATA(CURFLD)
349 GOSUB CHECK.ID ;* Check the newly entered ID handling index lookups
350 IF OK THEN
351 *
352 *****
353 *
354 * New ID (or result of index lookup) is good!
355 *
356 IDATA(CURFLD) = ID ;* Update the current field value in case of index
357 *
358 END ELSE
359 *
360 *****
361 *
362 * New ID is invalid
363 *
364 IDATA(CURFLD) = PREVAL ;* Restore previous value
365 NXTFLD = CURFLD ;* Reprompt
366 *
367 END
368 END
369 *
370 END CASE
371 END
372 *
373 WHILE CURFLD EQ NXTFLD DO REPEAT
374 *
375 * Redisplay the field data using the inactive highlighting
376 *
377 XLINE = CURFLD
378 ACTIVE = 0

```

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

325 END
326 IF IDATA(CURFLD) EQ SPACE(LEN(IDATA(CURFLD))) THEN IDATA(CURFLD) = ''
327 *
328 *****
329 *
330 * Perform field data validation if next field number changed
331 *
332 IF NXTFLD NE CURFLD THEN
333 BEGIN CASE
334 *
335 CASE CURFLD EQ 1
336 *
337 *****
338 *
339 * Validate the ID field. If NULL, quit. If changed, read new record.
340 *
341 IF CUST.ID EQ '' AND IDATA(CURFLD) EQ '' THEN
342 DONE = TRUE
343 XIT = TRUE
344 RETURN
345 END
346 *
347 IF IDATA(CURFLD) NE PREVAL THEN
348 ID = IDATA(CURFLD)
349 GOSUB CHECK.ID ;* Check the newly entered ID handling index lookups
350 IF OK THEN
351 *
352 *****
353 *
354 * New ID (or result of index lookup) is good!
355 *
356 IDATA(CURFLD) = ID ;* Update the current field value in case of index
357 *
358 END ELSE
359 *
360 *****
361 *
362 * New ID is invalid
363 *
364 IDATA(CURFLD) = PREVAL ;* Restore previous value
365 NXTFLD = CURFLD ;* Reprompt
366 *
367 END
368 END
369 *
370 END CASE
371 END
372 *
373 WHILE CURFLD EQ NXTFLD DO REPEAT
374 *
375 * Redisplay the field data using the inactive highlighting
376 *
377 XLINE = CURFLD
378 ACTIVE = 0

```

No differences found

Added(0,0)

Deleted(0,0)

Changed(0)

Changed in changed(0)

Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

379 GOSUB DSPLINE
380 *
381 RETURN
382 *
383 *
384 *****
385 *
386 * The CHECK.EXIT subroutine checks if any field data has changed, and
387 * prompts if the user wants to abandon changes. If no changes, or the user
388 * decides to abandon the changes, the DONE and XIT loop control variables
389 * are set to TRUE, causing all three loops to terminate, and the program
390 * itself to exit.
391 *
392 CHECK.EXIT: *
393 *
394 GOSUB CHECK.ABANDON
395 IF OK THEN
396   DONE = TRUE
397   XIT = TRUE
398 END
399 RETURN
400 *
401 *
402 *****
403 *
404 * The CHECK.ID subroutine validates a newly entered item-ID. If the
405 * current record has unsaved changes, the user is prompted to abandon the
406 * changes. If no changes, or the user abandons the changes, and the new ID
407 * is not NULL, an attempt is made to read a record using the new ID. If
408 * the read is not successful, the ID is assumed to be a search string, and
409 * the search subroutine is called to select an ID based on the search
410 * string. If a valid ID is returned (or if one was initially entered), the
411 * new record data is displayed. If the new ID is null, or if the search
412 * routine did not return a valid ID, a warning message is displayed and
413 * the OK indicator variable is set to FALSE. Otherwise it is set to TRUE.
414 *
415 CHECK.ID: *
416 *
417 *****
418 *
419 * Make sure we don't have any unsaved data before changing the ID
420 *
421 GOSUB CHECK.ABANDON
422 IF NOT(OK) THEN RETURN ;* Reprompt for the ID
423 IF ID EQ '' THEN
424   OK = FALSE ;* NULL is not a valid ID!
425 END ELSE
426 *
427 *****
428 *
429 * Check if user wants new ID
430 *
431 IF ID EQ 'N' OR ID EQ 'n' THEN
432   * Get next sequential ID

```

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

379 GOSUB DSPLINE
380 *
381 RETURN
382 *
383 *
384 *****
385 *
386 * The CHECK.EXIT subroutine checks if any field data has changed, and
387 * prompts if the user wants to abandon changes. If no changes, or the user
388 * decides to abandon the changes, the DONE and XIT loop control variables
389 * are set to TRUE, causing all three loops to terminate, and the program
390 * itself to exit.
391 *
392 CHECK.EXIT: *
393 *
394 GOSUB CHECK.ABANDON
395 IF OK THEN
396   DONE = TRUE
397   XIT = TRUE
398 END
399 RETURN
400 *
401 *
402 *****
403 *
404 * The CHECK.ID subroutine validates a newly entered item-ID. If the
405 * current record has unsaved changes, the user is prompted to abandon the
406 * changes. If no changes, or the user abandons the changes, and the new ID
407 * is not NULL, an attempt is made to read a record using the new ID. If
408 * the read is not successful, the ID is assumed to be a search string, and
409 * the search subroutine is called to select an ID based on the search
410 * string. If a valid ID is returned (or if one was initially entered), the
411 * new record data is displayed. If the new ID is null, or if the search
412 * routine did not return a valid ID, a warning message is displayed and
413 * the OK indicator variable is set to FALSE. Otherwise it is set to TRUE.
414 *
415 CHECK.ID: *
416 *
417 *****
418 *
419 * Make sure we don't have any unsaved data before changing the ID
420 *
421 GOSUB CHECK.ABANDON
422 IF NOT(OK) THEN RETURN ;* Reprompt for the ID
423 IF ID EQ '' THEN
424   OK = FALSE ;* NULL is not a valid ID!
425 END ELSE
426 *
427 *****
428 *
429 * Check if user wants new ID
430 *
431 IF ID EQ 'N' OR ID EQ 'n' THEN
432   * Get next sequential ID

```

No differences found

Added(0,0)

Deleted(0,0)

Changed(0)

Changed in changed(0)

Ignored


```

D:\Atwin32.dev\Samples\UIEX\UIEX2
433 READVU ID FROM FN.CUST.CTRL,'NEXT',2 THEN
434 WRITEV ID + 1 ON FN.CUST.CTRL,'NEXT',2
435 GOSUB RESTART
436 IDATA(1) = ID
437 END ELSE
438 PRINT EL:UNDREV:'Next item counter record not found!':NORMAL:BEL:
439 INPUT ANS:
440 PRINT EL:
441 OK = FALSE
442 RETURN
443 END
444 END ELSE
445 *
446 *****
447 *
448 * Try to read the customer record from the entered ID
449 *
450 GOSUB READ.RECORD
451 IF NOT(OK) THEN
452 *
453 *****
454 *
455 * The attempt to read a record failed - assume the ID is a search string
456 *
457 CALL UIEX.GET.CUST.IDX(XID,ID,FN.CUST.XREF,FN.CUST)
458 GOSUB DSPSCRN ;* Refresh the screen after index lookup
459 *
460 *****
461 *
462 * If the user did not select an item in the search routine, reprompt
463 *
464 IF XID EQ '' THEN RETURN
465 *
466 *****
467 *
468 * Try to read the customer record from the selected ID
469 *
470 ID = XID
471 GOSUB READ.RECORD
472 *
473 END
474 END
475 END
476 *
477 *****
478 *
479 * If success, display the new record, otherwise show warning message
480 *
481 IF OK THEN
482 GOSUB DSPDATA ;* Display new record
483 END ELSE
484 PRINT EL:UNDREV:'Please enter a valid customer ID!':NORMAL:BEL:
485 END
486 RETURN

```

```

D:\Atwin32.dev\Samples\UIEX\UIEX2
433 READVU ID FROM FN.CUST.CTRL,'NEXT',2 THEN
434 WRITEV ID + 1 ON FN.CUST.CTRL,'NEXT',2
435 GOSUB RESTART
436 IDATA(1) = ID
437 END ELSE
438 PRINT EL:UNDREV:'Next item counter record not found!':NORMAL:BEL:
439 INPUT ANS:
440 PRINT EL:
441 OK = FALSE
442 RETURN
443 END
444 END ELSE
445 *
446 *****
447 *
448 * Try to read the customer record from the entered ID
449 *
450 GOSUB READ.RECORD
451 IF NOT(OK) THEN
452 *
453 *****
454 *
455 * The attempt to read a record failed - assume the ID is a search string
456 *
457 CALL UIEX.GET.CUST.IDX(XID,ID,FN.CUST.XREF,FN.CUST)
458 GOSUB DSPSCRN ;* Refresh the screen after index lookup
459 *
460 *****
461 *
462 * If the user did not select an item in the search routine, reprompt
463 *
464 IF XID EQ '' THEN RETURN
465 *
466 *****
467 *
468 * Try to read the customer record from the selected ID
469 *
470 ID = XID
471 GOSUB READ.RECORD
472 *
473 END
474 END
475 END
476 *
477 *****
478 *
479 * If success, display the new record, otherwise show warning message
480 *
481 IF OK THEN
482 GOSUB DSPDATA ;* Display new record
483 END ELSE
484 PRINT EL:UNDREV:'Please enter a valid customer ID!':NORMAL:BEL:
485 END
486 RETURN

```

No differences found

Added(0,0)

Deleted(0,0)

Changed(0)

Changed in changed(0)

Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

487 *
488 *
489 *****
490 *
491 * The READ.RECORD subroutine reads a new customer record from the file and
492 * initializes the internal field data array (IDATA) from the record array
493 * (CUST.REC). If the record does not exist, the routine returns with the
494 * OK indicator variable set to FALSE. Otherwise OK is set to TRUE, and the
495 * CUST.ID variable is set to the new ID.
496 *
497 READ.RECORD: *
498 *
499 MATREAD CUST.REC FROM FN.CUST,ID THEN
500 CUST.ID = ID
501 IDATA(1) = CUST.ID
502 IDATA(2) = CUST.CONTACT
503 IDATA(3) = CUST.NAME
504 IDATA(4) = CUST.ADDRESS1
505 IDATA(5) = CUST.ADDRESS2
506 IDATA(6) = CUST.CITY
507 IDATA(7) = CUST.ST
508 IDATA(8) = CUST.ZIP
509 IDATA(9) = CUST.COUNTRY
510 IDATA(10) = CUST.PHONE
511 IDATA(11) = CUST.FAX
512 IDATA(12) = CUST.HISTORY
513 OK = TRUE ;* Set the SUCCESS indicator
514 END ELSE
515 OK = FALSE ;* Set the FAILURE indicator
516 END
517 RETURN
518 *
519 *
520 *****
521 *
522 * The DELETE.RECORD subroutine confirms that the user intends to delete
523 * the current record. If the action is confirmed, the CUST.DELETE
524 * subroutine is called to perform the deletion. A separate subroutine is
525 * used to handle updating indexes, etc.
526 *
527 DELETE.RECORD: *
528 *
529 IF CUST.ID NE '' THEN
530 PRINT EL:UNDREV:'Are you sure you want to delete this customer? ':NORMAL:
531 INPUT ANS:
532 IF ANS[1,1] EQ 'Y' OR ANS[1,1] EQ 'y' THEN
533 * Deletion has been confirmed - do the delete
534 OK = TRUE ;* Set the SUCCESS indicator
535 CALL UIEX.CUST.DELETE(CUST.ID, FN.CUST, FN.CUST.XREF)
536 DONE = TRUE ;* proceed to next record
537 END ELSE
538 OK = FALSE ;* Set the FAILURE indicator
539 END
540 END

```

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

487 *
488 *
489 *****
490 *
491 * The READ.RECORD subroutine reads a new customer record from the file and
492 * initializes the internal field data array (IDATA) from the record array
493 * (CUST.REC). If the record does not exist, the routine returns with the
494 * OK indicator variable set to FALSE. Otherwise OK is set to TRUE, and the
495 * CUST.ID variable is set to the new ID.
496 *
497 READ.RECORD: *
498 *
499 MATREAD CUST.REC FROM FN.CUST,ID THEN
500 CUST.ID = ID
501 IDATA(1) = CUST.ID
502 IDATA(2) = CUST.CONTACT
503 IDATA(3) = CUST.NAME
504 IDATA(4) = CUST.ADDRESS1
505 IDATA(5) = CUST.ADDRESS2
506 IDATA(6) = CUST.CITY
507 IDATA(7) = CUST.ST
508 IDATA(8) = CUST.ZIP
509 IDATA(9) = CUST.COUNTRY
510 IDATA(10) = CUST.PHONE
511 IDATA(11) = CUST.FAX
512 IDATA(12) = CUST.HISTORY
513 OK = TRUE ;* Set the SUCCESS indicator
514 END ELSE
515 OK = FALSE ;* Set the FAILURE indicator
516 END
517 RETURN
518 *
519 *
520 *****
521 *
522 * The DELETE.RECORD subroutine confirms that the user intends to delete
523 * the current record. If the action is confirmed, the CUST.DELETE
524 * subroutine is called to perform the deletion. A separate subroutine is
525 * used to handle updating indexes, etc.
526 *
527 DELETE.RECORD: *
528 *
529 IF CUST.ID NE '' THEN
530 PRINT EL:UNDREV:'Are you sure you want to delete this customer? ':NORMAL:
531 INPUT ANS:
532 IF ANS[1,1] EQ 'Y' OR ANS[1,1] EQ 'y' THEN
533 * Deletion has been confirmed - do the delete
534 OK = TRUE ;* Set the SUCCESS indicator
535 CALL UIEX.CUST.DELETE(CUST.ID, FN.CUST, FN.CUST.XREF)
536 DONE = TRUE ;* proceed to next record
537 END ELSE
538 OK = FALSE ;* Set the FAILURE indicator
539 END
540 END

```

No differences found

Added(0,0)

Deleted(0,0)

Changed(0)

Changed in changed(0)

Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

541 RETURN
542 *
543 *
544 *****
545 *
546 * The SAVE.RECORD subroutine copies internal field data from the IDATA
547 * array to the customer record array (CUST.REC). The CUST.UPDATE
548 * subroutine is called to perform the update. A separate subroutine is
549 * used to handle updating indexes, etc.
550 *
551 SAVE.RECORD: *
552 *
553 IF IDATA(1) NE '' THEN
554 * Copy data from the internal field data array (IDATA) to the CUST.REC array
555 CUST.ID = IDATA(1)
556 CUST.CONTACT = IDATA(2)
557 CUST.NAME = IDATA(3)
558 CUST.ADDRESS1 = IDATA(4)
559 CUST.ADDRESS2 = IDATA(5)
560 CUST.CITY = IDATA(6)
561 CUST.ST = IDATA(7)
562 CUST.ZIP = IDATA(8)
563 CUST.COUNTRY = IDATA(9)
564 CUST.PHONE = IDATA(10)
565 CUST.FAX = IDATA(11)
566 CUST.HISTORY = IDATA(12)
567 * Update the file
568 CALL UIEX.CUST.UPDATE(CUST.ID,MAT CUST.REC, FN.CUST, FN.CUST.XREF)
569 OK = TRUE ;* Set the SUCCESS indicator
570 END ELSE
571 OK = FALSE ;* Set the FAILURE indicator
572 END
573 DONE = TRUE ;* proceed to next record
574 RETURN
575 *
576 *
577 *****
578 *
579 * The RESTART subroutine prepares the internal field data array (IDATA),
580 * customer record array (CUST.REC) and ID for a new customer record.
581 *
582 RESTART: *
583 *
584 CUST.ID = ''
585 MAT CUST.REC = ''
586 MAT IDATA = ''
587 ACTIVE = 0
588 RETURN
589 *
590 *
591 *****
592 *
593 * The CHECK.CHANGED subroutine checks if any internal field data has been
594 * changed. The OK indicator variable is set to FALSE if any data is

```

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

541 RETURN
542 *
543 *
544 *****
545 *
546 * The SAVE.RECORD subroutine copies internal field data from the IDATA
547 * array to the customer record array (CUST.REC). The CUST.UPDATE
548 * subroutine is called to perform the update. A separate subroutine is
549 * used to handle updating indexes, etc.
550 *
551 SAVE.RECORD: *
552 *
553 IF IDATA(1) NE '' THEN
554 * Copy data from the internal field data array (IDATA) to the CUST.REC array
555 CUST.ID = IDATA(1)
556 CUST.CONTACT = IDATA(2)
557 CUST.NAME = IDATA(3)
558 CUST.ADDRESS1 = IDATA(4)
559 CUST.ADDRESS2 = IDATA(5)
560 CUST.CITY = IDATA(6)
561 CUST.ST = IDATA(7)
562 CUST.ZIP = IDATA(8)
563 CUST.COUNTRY = IDATA(9)
564 CUST.PHONE = IDATA(10)
565 CUST.FAX = IDATA(11)
566 CUST.HISTORY = IDATA(12)
567 * Update the file
568 CALL UIEX.CUST.UPDATE(CUST.ID,MAT CUST.REC, FN.CUST, FN.CUST.XREF)
569 OK = TRUE ;* Set the SUCCESS indicator
570 END ELSE
571 OK = FALSE ;* Set the FAILURE indicator
572 END
573 DONE = TRUE ;* proceed to next record
574 RETURN
575 *
576 *
577 *****
578 *
579 * The RESTART subroutine prepares the internal field data array (IDATA),
580 * customer record array (CUST.REC) and ID for a new customer record.
581 *
582 RESTART: *
583 *
584 CUST.ID = ''
585 MAT CUST.REC = ''
586 MAT IDATA = ''
587 ACTIVE = 0
588 RETURN
589 *
590 *
591 *****
592 *
593 * The CHECK.CHANGED subroutine checks if any internal field data has been
594 * changed. The OK indicator variable is set to FALSE if any data is

```

No differences found

Added(0,0)

Deleted(0,0)

Changed(0)

Changed in changed(0)

Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

595 * changed, otherwise it is set to TRUE. The ID field is not checked, since
596 * it is appropriately handled by the CHECK.ID subroutine.
597 *
598 CHECK.CHANGED: *
599 *
600 OK = FALSE
601 IF CUST.CONTACT # IDATA(2) THEN RETURN
602 IF CUST.NAME # IDATA(3) THEN RETURN
603 IF CUST.ADDRESS1 # IDATA(4) THEN RETURN
604 IF CUST.ADDRESS2 # IDATA(5) THEN RETURN
605 IF CUST.CITY # IDATA(6) THEN RETURN
606 IF CUST.ST # IDATA(7) THEN RETURN
607 IF CUST.ZIP # IDATA(8) THEN RETURN
608 IF CUST.COUNTRY # IDATA(9) THEN RETURN
609 IF CUST.PHONE # IDATA(10) THEN RETURN
610 IF CUST.FAX # IDATA(11) THEN RETURN
611 IF CUST.HISTORY # IDATA(12) THEN RETURN
612 OK = TRUE
613 RETURN
614 *
615 *
616 *****
617 *
618 * The CHECK.ABANDON subroutine calls CHECK.CHANGED. If any changes are
619 * found, a message is displayed and the user is prompted to abandon the
620 * changes. If there are no changes, or if the user decides to abandon
621 * changes, the OK indicator variable is set to TRUE. Otherwise it is set
622 * to FALSE.
623 *
624 CHECK.ABANDON: *
625 *
626 GOSUB CHECK.CHANGED
627 IF NOT(OK) THEN
628   PRINT EL:UNDREV:'Do you want to abandon all your changes? ':NORMAL:BEL:
629   INPUT ANS:
630   IF ANS[1,1] EQ 'Y' OR ANS[1,1] EQ 'y' THEN OK = 1
631   PRINT EL:
632 END
633 RETURN
634 *
635 *
636 *****
637 *
638 * The DSPSCRN subroutine is used to refresh the entire screen.
639 *
640 DSPSCRN: *
641 *
642 * Clear the screen
643 PRINT CLR:NORMAL:
644 *
645 * Display heading & labels
646 PRINT @(5,0):'Customer File Maintenance':
647 FOR XLINE = 1 TO NUMFLDS
648   LBWD = CONTROL(XLINE)<1,4>

```

D:\Atwin32.dev\Samples\UIEX\UIEX2

```

595 * changed, otherwise it is set to TRUE. The ID field is not checked, since
596 * it is appropriately handled by the CHECK.ID subroutine.
597 *
598 CHECK.CHANGED: *
599 *
600 OK = FALSE
601 IF CUST.CONTACT # IDATA(2) THEN RETURN
602 IF CUST.NAME # IDATA(3) THEN RETURN
603 IF CUST.ADDRESS1 # IDATA(4) THEN RETURN
604 IF CUST.ADDRESS2 # IDATA(5) THEN RETURN
605 IF CUST.CITY # IDATA(6) THEN RETURN
606 IF CUST.ST # IDATA(7) THEN RETURN
607 IF CUST.ZIP # IDATA(8) THEN RETURN
608 IF CUST.COUNTRY # IDATA(9) THEN RETURN
609 IF CUST.PHONE # IDATA(10) THEN RETURN
610 IF CUST.FAX # IDATA(11) THEN RETURN
611 IF CUST.HISTORY # IDATA(12) THEN RETURN
612 OK = TRUE
613 RETURN
614 *
615 *
616 *****
617 *
618 * The CHECK.ABANDON subroutine calls CHECK.CHANGED. If any changes are
619 * found, a message is displayed and the user is prompted to abandon the
620 * changes. If there are no changes, or if the user decides to abandon
621 * changes, the OK indicator variable is set to TRUE. Otherwise it is set
622 * to FALSE.
623 *
624 CHECK.ABANDON: *
625 *
626 GOSUB CHECK.CHANGED
627 IF NOT(OK) THEN
628   PRINT EL:UNDREV:'Do you want to abandon all your changes? ':NORMAL:BEL:
629   INPUT ANS:
630   IF ANS[1,1] EQ 'Y' OR ANS[1,1] EQ 'y' THEN OK = 1
631   PRINT EL:
632 END
633 RETURN
634 *
635 *
636 *****
637 *
638 * The DSPSCRN subroutine is used to refresh the entire screen.
639 *
640 DSPSCRN: *
641 *
642 * Clear the screen
643 PRINT CLR:NORMAL:
644 *
645 * Display heading & labels
646 PRINT @(5,0):'Customer File Maintenance':
647 FOR XLINE = 1 TO NUMFLDS
648   LBWD = CONTROL(XLINE)<1,4>

```

No differences found

Added(0,0)

Deleted(0,0)

Changed(0)

Changed in changed(0)

Ignored

D:\Atwin32.dev\Samples\UIEX\UIEX2	D:\Atwin32.dev\Samples\UIEX\UIEX2
649 LBTX = (XLINE 'L#2 ') : CONTROL(XLINE)<1,1> ;* Prepend line number to label	649 LBTX = (XLINE 'L#2 ') : CONTROL(XLINE)<1,1> ;* Prepend line number to label
650 IF LBWD > 0 THEN LBTX = (LBTX : STR('.',LBWD))[1,LBWD] ;* Pad label with dots	650 IF LBWD > 0 THEN LBTX = (LBTX : STR('.',LBWD))[1,LBWD] ;* Pad label with dots
651 PRINT @(CONTROL(XLINE)<1,2>,CONTROL(XLINE)<1,3>):LBTX:	651 PRINT @(CONTROL(XLINE)<1,2>,CONTROL(XLINE)<1,3>):LBTX:
652 NEXT XLINE	652 NEXT XLINE
653 *	653 *
654 * Display the field data	654 * Display the field data
655 GOSUB DSPDATA	655 GOSUB DSPDATA
656 RETURN	656 RETURN
657 *	657 *
658 *	658 *
659 *****	659 *****
660 *	660 *
661 * The DSPDATA subroutine is used to refresh the field data for all fields.	661 * The DSPDATA subroutine is used to refresh the field data for all fields.
662 *	662 *
663 DSPDATA: *	663 DSPDATA: *
664 *	664 *
665 FOR XLINE = 1 TO NUMFLDS	665 FOR XLINE = 1 TO NUMFLDS
666 GOSUB DSPLINE	666 GOSUB DSPLINE
667 NEXT XLINE	667 NEXT XLINE
668 RETURN	668 RETURN
669 *	669 *
670 *	670 *
671 *****	671 *****
672 *	672 *
673 * The DSPLINE subroutine is used to refresh the field data for one field.	673 * The DSPLINE subroutine is used to refresh the field data for one field.
674 * The field to be refreshed is specified by the XLINE variable. If the	674 * The field to be refreshed is specified by the XLINE variable. If the
675 * field is active (XLINE = ACTIVE), then the field data is displayed using	675 * field is active (XLINE = ACTIVE), then the field data is displayed using
676 * the REVERSE display attribute. Otherwise it is displayed using the	676 * the REVERSE display attribute. Otherwise it is displayed using the
677 * DIMREV display attribute.	677 * DIMREV display attribute.
678 *	678 *
679 DSPLINE: *	679 DSPLINE: *
680 *	680 *
681 MSK = 'L#':CONTROL(XLINE)<1,7>	681 MSK = 'L#':CONTROL(XLINE)<1,7>
682 PRINT @(CONTROL(XLINE)<1,5>,CONTROL(XLINE)<1,6>):	682 PRINT @(CONTROL(XLINE)<1,5>,CONTROL(XLINE)<1,6>):
683 IF XLINE EQ ACTIVE THEN	683 IF XLINE EQ ACTIVE THEN
684 PRINT REVERSE:	684 PRINT REVERSE:
685 END ELSE	685 END ELSE
686 PRINT DIMREV:	686 PRINT DIMREV:
687 END	687 END
688 PRINT IDATA(XLINE) MSK:	688 PRINT IDATA(XLINE) MSK:
689 PRINT NORMAL:	689 PRINT NORMAL:
690 RETURN	690 RETURN
691 *	691 *
692 END	692 END
693	693

No differences found

Added(0,0)

Deleted(0,0)

Changed(0)

Changed in changed(0)

Ignored