

```

0001 *
0002 *****
0003 * Copyright (c) 2004 Zumasy, Inc. as an unpublished work. Permission is *
0004 * hereby granted, free of charge, to any person obtaining a copy of this *
0005 * software, to use the software without restriction, including without *
0006 * limitation the rights to use, copy, modify, merge, publish or *
0007 * distribute the software, and to permit persons to whom the software is *
0008 * furnished to do so, subject to the following conditions: This *
0009 * copyright notice and permission notice shall be included in all copies *
0010 * or substantial portions of the software. THE SOFTWARE IS PROVIDED "AS *
0011 * IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT *
0012 * NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A *
0013 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE *
0014 * COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, *
0015 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT *
0016 * OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN *
0017 * THE SOFTWARE. *
0018 *****
0019 *****
0020 *
0021 * USER INTERFACE EXAMPLE PROGRAM 2
0022 *
0023 *****
0024 *****
0025 *
0026 * This example illustrates a character-based interface which uses AccuTerm
0027 * Visual Styles to display screen elements using a Windows-like look (if
0028 * available). In this example, Visual Styles (border effects and colors)
0029 * are associated with certain display attributes. The program will run
0030 * without Visual Styles if used with dumb terminals, older versions of
0031 * AccuTerm or other terminal emulators.
0032 *
0033 *****
0034 *
0035 * The CUST.REC INCLUDE item declares the CUST.REC array, which is used to
0036 * store a working copy of the current data record. It also defines the
0037 * record layout by equating field names to array positions in the CUST.REC
0038 * array.
0039 *
0040 $INCLUDE UIEX.CUST.REC
0041 *
0042 *****
0043 *
0044 * EQUates for control characters and delimiters and other constants
0045 *
0046 EQU VM TO CHAR(253)
0047 EQU BEL TO CHAR(7)
0048 EQU ESC TO CHAR(27)
0049 EQU FALSE TO 0
0050 EQU TRUE TO 1
0051 *
0052 *****
0053 *
0054 * Open our files...
0055 *
0056 OPEN 'CUST.SAMPLE' TO FN.CUST ELSE PRINT 'NO CUST.SAMPLE FILE'; STOP
0057 OPEN 'CUST.SAMPLE.CTRL' TO FN.CUST.CTRL ELSE PRINT 'NO CUST.SAMPLE.CTRL FILE'; STOP
0058 OPEN 'CUST.SAMPLE.XREF' TO FN.CUST.XREF ELSE PRINT 'NO CUST.SAMPLE.XREF'; STOP
0059 *
0060 *****
0061 *

```

```
0062 * Define the data field control structures. NUMFLDS is the number of
0063 * fields. The CONTROL array defines the field control elements such as
0064 * field label, label position, field position and size, and field prompt.
0065 * The IDATA array stores the current field values, and is initialized from
0066 * the CUST.REC array when a data record is read from the file. When the
0067 * record is updated, values are copied from the IDATA array to the
0068 * CUST.REC array and the CUST.REC array is passed to the CUST.UPDATE
0069 * subroutine to update the data and index files (a separate subroutine is
0070 * used to update the file since the update process may handle other
0071 * functions like updating indexes).
0072 *
0073 NUMFLDS = 12
0074 DIM CONTROL(12)
0075 DIM IDATA(12)
0076 *
0077 * Define field control elements
0078 * value 1: field label text
0079 * value 2: field label column
0080 * value 3: field label row
0081 * value 4: field label width (0 for actual width, no padding)
0082 * value 5: field data column
0083 * value 6: field data row
0084 * value 7: field data width
0085 * value 8: field data height
0086 * value 9: field prompt message
0087 CONTROL(1) = 'Customer ID???0?8??0?]Enter the ID, or name to search for, or N for
next number'
0088 CONTROL(2) = 'Contact???0?8??0?]Enter the contact name'
0089 CONTROL(3) = 'Company name???0?8??0?]Enter the company name'
0090 CONTROL(4) = 'Address line 1???0?8??0?]Enter address line 1'
0091 CONTROL(5) = 'Address line 2???0?8??0?]Enter address line 2'
0092 CONTROL(6) = 'City???0?8??5?]Enter the city'
0093 CONTROL(7) = 'State or province???0?8??5?]Enter the state or province abbreviation'
0094 CONTROL(8) = 'Zip/postal code??0?0?8?0?0?]Enter the zip or postal code'
0095 CONTROL(9) = 'Country??1?0?8?1?8?]Enter the country'
0096 CONTROL(10) = 'Phone??2?0?8?2?5?]Enter the phone number'
0097 CONTROL(11) = 'Fax??3?0?8?3?5?]Enter the fax number'
0098 CONTROL(12) = 'Notes??4?0?8?4?0?]Enter any notes about this customer'
0099 *
0100 *****
0101 *
0102 * Define some screen control strings for prompts & errors
0103 *
0104 PROMPT ''
0105 CLR = @(-1) ;* Clear entire screen
0106 CEOL = @(-4) ;* Clear to end of line
0107 PL = @(5,22):CEOL ;* Prompt line
0108 EL = @(5,23):CEOL ;* Error line
0109 *
0110 * Wyse 60 attributes for NORMAL, REVERSE, DIM REVERSE and UNDERLINE
0111 * REVERSE (we like Wyse 60 because the attributes dont take any space on
0112 * the screen, and more than one attribute can be displayed at one time).
0113 * If running AccuTerm in Viewpoint A2 Enhanced emulation, you can use the
0114 * ADDS 4000 attributes instead. Just change the upper-case "G" below to
0115 * lower-case "g".
0116 *
0117 NORMAL = ESC:'G0' ;* Normal - display headings & field labels
0118 REVERSE = ESC:'G4' ;* Reverse - display active field data
0119 DIMREV = ESC:'Gt' ;* Dim Reverse - display inactive field data
0120 UNDREV = ESC:'G<' ;* Underline Reverse - display warnings
0121 *
```

```
0122 *****
0123 *
0124 * This program processes one customer record at a time, and is organized
0125 * using three nested loops. The outermost loop (the RECORD loop) is
0126 * executed once for each record accessed. The loop repeats until the XIT
0127 * control variable is set to TRUE.
0128 *
0129 XIT = FALSE
0130 LOOP UNTIL XIT DO
0131 *
0132 *****
0133 *
0134 * Before prompting for the customer ID, reset the internal data array
0135 * (IDATA) and CUST.ID variables, then clear the screen and display the
0136 * heading and field labels.
0137 *
0138 GOSUB RESTART
0139 GOSUB DSPSCRN
0140 *
0141 *****
0142 *
0143 * The middle loop is the ACTION loop. It is executed for the current
0144 * record until the user performs an action that terminates processing of
0145 * that record, such as exiting, cancelling, saving or deleting the
0146 * record. The field number to begin prompting (NXTFLD) is initialized to
0147 * 1, causing the ID field to be prompted first. The loop immediately
0148 * enters the PROMPT loop, followed by the field modification prompt. The
0149 * loop repeats until the DONE control variable is set to TRUE.
0150 *
0151 NXTFLD = 1
0152 DONE = FALSE
0153 LOOP
0154 *
0155 *****
0156 *
0157 * The inner loop is the PROMPT loop. It is executed for each prompt
0158 * field as specified by the NXTFLD variable. The prompt loop simply
0159 * calls the local INPUT.FIELD subroutine which performs field input,
0160 * data validation and keyboard command decoding. The FIELD loop repeats
0161 * until the field number is greater than the number of fields, or until
0162 * the DONE control variable is set to TRUE. The DONE control variable
0163 * may be set to TRUE in the INPUT.FIELD subroutine, if the user enters a
0164 * NULL to for the item-ID.
0165 *
0166 LOOP
0167     CURFLD = NXTFLD
0168 UNTIL DONE OR CURFLD > NUMFLDS DO
0169 *
0170 *****
0171 *
0172 * Prompt for the next field. Update the NXTFLD variable with the field
0173 * number to prompt next.
0174 *
0175     GOSUB INPUT.FIELD
0176 REPEAT ;* end of PROMPT loop
0177 *
0178 *****
0179 *
0180 * If the FIELD loop exited with the DONE control variable set to TRUE,
0181 * bypass the modification prompt because no action is required.
0182 * Otherwise, prompt for which field to modify, or other actions such as
```

```

0183 * save or delete.
0184 *
0185 IF NOT(DONE) THEN
0186 *
0187 NXTFLD = NUMFLDS + 1 ;* assume we need to reprompt for action
0188 *
0189 PRINT PL:'Enter field number to modify or FI to save, DE to delete, EX to exit: ':
0190 INPUT ANS:
0191 PRINT EL:
0192 *
0193 *****
0194 *
0195 * Decode the response
0196 *
0197 ANS = OCONV(ANS, 'MCU')
0198 BEGIN CASE
0199 CASE NUM(ANS) AND ANS >= 1 AND ANS <= NUMFLDS; NXTFLD = ANS
0200 CASE ANS EQ 'FI'; GOSUB SAVE.RECORD
0201 CASE ANS EQ 'DE'; GOSUB DELETE.RECORD
0202 CASE ANS EQ 'EX' OR ANS EQ ''; GOSUB CHECK.EXIT
0203 END CASE
0204 *
0205 END
0206 *
0207 UNTIL DONE DO REPEAT ;* end of ACTION loop
0208 *
0209 REPEAT ;* end of RECORD loop
0210 *
0211 *****
0212 *
0213 * All done - clear the screen and exit!
0214 PRINT CLR:
0215 STOP
0216 *
0217 *
0218 *****
0219 *****
0220 * LOCAL SUBROUTINES
0221 *****
0222 *****
0223 *
0224 *
0225 *****
0226 *
0227 * The INPUT.FIELD subroutine is the main prompting routine. This routine
0228 * displays the prompt string (from the CONTROL array), and enters a loop,
0229 * prompting for a specified field (CURFLD) and setting the next field
0230 * variable (NXTFLD) appropriately. The loop repeats until the NXTFLD
0231 * (initially set to CURFLD) changes. This causes the field prompt to be
0232 * repeated in case invalid data is entered (illegal customer ID, etc.) If
0233 * a NULL is entered for the ID field, and there is no current record, the
0234 * ACTION loop control variable, DONE, and the RECORD loop control
0235 * variable, XIT, are set to TRUE, and this routine exits.
0236 *
0237 INPUT.FIELD:
0238 *
0239 *****
0240 *
0241 * Display the prompt message
0242 *
0243 PRINT PL:CONTROL(CURFLD)<1,9>:

```

```

0244 *
0245 *****
0246 *
0247 * Initialize current value, next field
0248 *
0249 PREVAL = IDATA(CURFLD) ;* Save previous field value to detect change in value
0250 *
0251 *****
0252 *
0253 * Prompt for this field until NXTFLD variable is updated
0254 *
0255 LOOP
0256 *
0257 *****
0258 *
0259 * Assume next field number is next sequential field
0260 *
0261 NXTFLD = CURFLD + 1
0262 *
0263 * Highlight current field data
0264 *
0265 XLINE = CURFLD ;* Set the field number to display
0266 ACTIVE = CURFLD ;* Set the active field number to highlight field
0267 GOSUB DSPLINE
0268 *
0269 * Prompt for input
0270 *
0271 PRINT @(CONTROL(CURFLD)<1,5>,CONTROL(CURFLD)<1,6>):REVERSE:
0272 INPUT IDATA(CURFLD):
0273 *
0274 * Check for NULL
0275 *
0276 K = LEN(IDATA(CURFLD))
0277 IF K = 0 THEN
0278 *
0279 * No change if NULL entered
0280 *
0281 IDATA(CURFLD) = PREVAL
0282 END ELSE
0283 *
0284 * Erase old data
0285 *
0286 IF K < CONTROL(CURFLD)<1,7> THEN
0287 PRINT @(K+CONTROL(CURFLD)<1,5>,CONTROL(CURFLD)<1,6>):SPACE(CONTROL(CURFLD)<1,7> -
K):
0288 END
0289 END
0290 *
0291 * Reset display attribute
0292 *
0293 PRINT NORMAL:
0294 *
0295 * Clear the error line
0296 *
0297 PRINT EL:
0298 *
0299 *****
0300 *
0301 * Check for any special values (like 'END' or 'EXIT')
0302 *
0303 IF OCONV(IDATA(CURFLD),'MCU') EQ 'END' THEN

```

```

0304 IDATA(CURFLD) = PREVAL ;* Restore previous value
0305 GOSUB CHECK.ABANDON ;* Ensure OK to loose changes
0306 IF OK THEN
0307 *
0308 *****
0309 *
0310 * No changes, or user OKs abandoning them, so we are outa here!
0311 *
0312 DONE = TRUE
0313 XIT = TRUE
0314 RETURN
0315 *
0316 END ELSE
0317 *
0318 *****
0319 *
0320 * User does not want to abandon changes, so reprompt
0321 *
0322 NXTFLD = CURFLD ;* Reprompt
0323 *
0324 END
0325 END
0326 IF IDATA(CURFLD) EQ SPACE(LEN(IDATA(CURFLD))) THEN IDATA(CURFLD) = ''
0327 *
0328 *****
0329 *
0330 * Perform field data validation if next field number changed
0331 *
0332 IF NXTFLD NE CURFLD THEN
0333 BEGIN CASE
0334 *
0335 CASE CURFLD EQ 1
0336 *
0337 *****
0338 *
0339 * Validate the ID field. If NULL, quit. If changed, read new record.
0340 *
0341 IF CUST.ID EQ '' AND IDATA(CURFLD) EQ '' THEN
0342 DONE = TRUE
0343 XIT = TRUE
0344 RETURN
0345 END
0346 *
0347 IF IDATA(CURFLD) NE PREVAL THEN
0348 ID = IDATA(CURFLD)
0349 GOSUB CHECK.ID ;* Check the newly entered ID handling index lookups
0350 IF OK THEN
0351 *
0352 *****
0353 *
0354 * New ID (or result of index lookup) is good!
0355 *
0356 IDATA(CURFLD) = ID ;* Update the current field value in case of index lookup
0357 *
0358 END ELSE
0359 *
0360 *****
0361 *
0362 * New ID is invalid
0363 *
0364 IDATA(CURFLD) = PREVAL ;* Restore previous value

```

```
0365     NXTFLD = CURFLD ;* Reprompt
0366     *
0367     END
0368     END
0369     *
0370     END CASE
0371     END
0372     *
0373     WHILE CURFLD EQ NXTFLD DO REPEAT
0374     *
0375     * Redisplay the field data using the inactive highlighting
0376     *
0377     XLINE = CURFLD
0378     ACTIVE = 0
0379     GOSUB DSPLINE
0380     *
0381     RETURN
0382     *
0383     *
0384     *****
0385     *
0386     * The CHECK.EXIT subroutine checks if any field data has changed, and
0387     * prompts if the user wants to abandon changes. If no changes, or the user
0388     * decides to abandon the changes, the DONE and XIT loop control variables
0389     * are set to TRUE, causing all three loops to terminate, and the program
0390     * itself to exit.
0391     *
0392     CHECK.EXIT: *
0393     *
0394     GOSUB CHECK.ABANDON
0395     IF OK THEN
0396         DONE = TRUE
0397         XIT = TRUE
0398     END
0399     RETURN
0400     *
0401     *
0402     *****
0403     *
0404     * The CHECK.ID subroutine validates a newly entered item-ID. If the
0405     * current record has unsaved changes, the user is prompted to abandon the
0406     * changes. If no changes, or the user abandons the changes, and the new ID
0407     * is not NULL, an attempt is made to read a record using the new ID. If
0408     * the read is not successful, the ID is assumed to be a search string, and
0409     * the search subroutine is called to select an ID based on the search
0410     * string. If a valid ID is returned (or if one was initially entered), the
0411     * new record data is displayed. If the new ID is null, or if the search
0412     * routine did not return a valid ID, a warning message is displayed and
0413     * the OK indicator variable is set to FALSE. Otherwise it is set to TRUE.
0414     *
0415     CHECK.ID: *
0416     *
0417     *****
0418     *
0419     * Make sure we don't have any unsaved data before changing the ID
0420     *
0421     GOSUB CHECK.ABANDON
0422     IF NOT(OK) THEN RETURN ;* Reprompt for the ID
0423     IF ID EQ '' THEN
0424         OK = FALSE ;* NULL is not a valid ID!
0425     END ELSE
```

```

0426 *
0427 *****
0428 *
0429 * Check if user wants new ID
0430 *
0431 IF ID EQ 'N' OR ID EQ 'n' THEN
0432 * Get next sequential ID
0433 READVU ID FROM FN.CUST.CTRL, 'NEXT', 2 THEN
0434 WRITEV ID + 1 ON FN.CUST.CTRL, 'NEXT', 2
0435 GOSUB RESTART
0436 IDATA(1) = ID
0437 END ELSE
0438 PRINT EL:UNDREV:'Next item counter record not found!':NORMAL:BEL:
0439 INPUT ANS:
0440 PRINT EL:
0441 OK = FALSE
0442 RETURN
0443 END
0444 END ELSE
0445 *
0446 *****
0447 *
0448 * Try to read the customer record from the entered ID
0449 *
0450 GOSUB READ.RECORD
0451 IF NOT(OK) THEN
0452 *
0453 *****
0454 *
0455 * The attempt to read a record failed - assume the ID is a search string
0456 *
0457 CALL UIEX.GET.CUST.IDX(XID, ID, FN.CUST.XREF, FN.CUST)
0458 GOSUB DSPSCRN ; * Refresh the screen after index lookup
0459 *
0460 *****
0461 *
0462 * If the user did not select an item in the search routine, reprompt
0463 *
0464 IF XID EQ '' THEN RETURN
0465 *
0466 *****
0467 *
0468 * Try to read the customer record from the selected ID
0469 *
0470 ID = XID
0471 GOSUB READ.RECORD
0472 *
0473 END
0474 END
0475 END
0476 *
0477 *****
0478 *
0479 * If success, display the new record, otherwise show warning message
0480 *
0481 IF OK THEN
0482 GOSUB DSPDATA ; * Display new record
0483 END ELSE
0484 PRINT EL:UNDREV:'Please enter a valid customer ID!':NORMAL:BEL:
0485 END
0486 RETURN

```



```

0487 *
0488 *
0489 *****
0490 *
0491 * The READ.RECORD subroutine reads a new customer record from the file and
0492 * initializes the internal field data array (IDATA) from the record array
0493 * (CUST.REC). If the record does not exist, the routine returns with the
0494 * OK indicator variable set to FALSE. Otherwise OK is set to TRUE, and the
0495 * CUST.ID variable is set to the new ID.
0496 *
0497 READ.RECORD: *
0498 *
0499 MATREAD CUST.REC FROM FN.CUST, ID THEN
0500     CUST.ID = ID
0501     IDATA(1) = CUST.ID
0502     IDATA(2) = CUST.CONTACT
0503     IDATA(3) = CUST.NAME
0504     IDATA(4) = CUST.ADDRESS1
0505     IDATA(5) = CUST.ADDRESS2
0506     IDATA(6) = CUST.CITY
0507     IDATA(7) = CUST.ST
0508     IDATA(8) = CUST.ZIP
0509     IDATA(9) = CUST.COUNTRY
0510     IDATA(10) = CUST.PHONE
0511     IDATA(11) = CUST.FAX
0512     IDATA(12) = CUST.HISTORY
0513     OK = TRUE ;* Set the SUCCESS indicator
0514 END ELSE
0515     OK = FALSE ;* Set the FAILURE indicator
0516 END
0517 RETURN
0518 *
0519 *
0520 *****
0521 *
0522 * The DELETE.RECORD subroutine confirms that the user intends to delete
0523 * the current record. If the action is confirmed, the CUST.DELETE
0524 * subroutine is called to perform the deletion. A separate subroutine is
0525 * used to handle updating indexes, etc.
0526 *
0527 DELETE.RECORD: *
0528 *
0529 IF CUST.ID NE '' THEN
0530     PRINT EL:UNDREV:'Are you sure you want to delete this customer? ':NORMAL:
0531     INPUT ANS:
0532     IF ANS[1,1] EQ 'Y' OR ANS[1,1] EQ 'y' THEN
0533         * Deletion has been confirmed - do the delete
0534         OK = TRUE ;* Set the SUCCESS indicator
0535         CALL UIEX.CUST.DELETE(CUST.ID, FN.CUST, FN.CUST.XREF)
0536         DONE = TRUE ;* proceed to next record
0537     END ELSE
0538         OK = FALSE ;* Set the FAILURE indicator
0539     END
0540 END
0541 RETURN
0542 *
0543 *
0544 *****
0545 *
0546 * The SAVE.RECORD subroutine copies internal field data from the IDATA
0547 * array to the customer record array (CUST.REC). The CUST.UPDATE

```

```
0548 * subroutine is called to perform the update. A separate subroutine is
0549 * used to handle updating indexes, etc.
0550 *
0551 SAVE.RECORD: *
0552 *
0553 IF IDATA(1) NE '' THEN
0554 * Copy data from the internal field data array (IDATA) to the CUST.REC array
0555 CUST.ID = IDATA(1)
0556 CUST.CONTACT = IDATA(2)
0557 CUST.NAME = IDATA(3)
0558 CUST.ADDRESS1 = IDATA(4)
0559 CUST.ADDRESS2 = IDATA(5)
0560 CUST.CITY = IDATA(6)
0561 CUST.ST = IDATA(7)
0562 CUST.ZIP = IDATA(8)
0563 CUST.COUNTRY = IDATA(9)
0564 CUST.PHONE = IDATA(10)
0565 CUST.FAX = IDATA(11)
0566 CUST.HISTORY = IDATA(12)
0567 * Update the file
0568 CALL UIEX.CUST.UPDATE(CUST.ID,MAT CUST.REC, FN.CUST, FN.CUST.XREF)
0569 OK = TRUE ;* Set the SUCCESS indicator
0570 END ELSE
0571 OK = FALSE ;* Set the FAILURE indicator
0572 END
0573 DONE = TRUE ;* proceed to next record
0574 RETURN
0575 *
0576 *
0577 *****
0578 *
0579 * The RESTART subroutine prepares the internal field data array (IDATA),
0580 * customer record array (CUST.REC) and ID for a new customer record.
0581 *
0582 RESTART: *
0583 *
0584 CUST.ID = ''
0585 MAT CUST.REC = ''
0586 MAT IDATA = ''
0587 ACTIVE = 0
0588 RETURN
0589 *
0590 *
0591 *****
0592 *
0593 * The CHECK.CHANGED subroutine checks if any internal field data has been
0594 * changed. The OK indicator variable is set to FALSE if any data is
0595 * changed, otherwise it is set to TRUE. The ID field is not checked, since
0596 * it is appropriately handled by the CHECK.ID subroutine.
0597 *
0598 CHECK.CHANGED: *
0599 *
0600 OK = FALSE
0601 IF CUST.CONTACT # IDATA(2) THEN RETURN
0602 IF CUST.NAME # IDATA(3) THEN RETURN
0603 IF CUST.ADDRESS1 # IDATA(4) THEN RETURN
0604 IF CUST.ADDRESS2 # IDATA(5) THEN RETURN
0605 IF CUST.CITY # IDATA(6) THEN RETURN
0606 IF CUST.ST # IDATA(7) THEN RETURN
0607 IF CUST.ZIP # IDATA(8) THEN RETURN
0608 IF CUST.COUNTRY # IDATA(9) THEN RETURN
```

```
0609 IF CUST.PHONE # IDATA(10) THEN RETURN
0610 IF CUST.FAX # IDATA(11) THEN RETURN
0611 IF CUST.HISTORY # IDATA(12) THEN RETURN
0612 OK = TRUE
0613 RETURN
0614 *
0615 *
0616 *****
0617 *
0618 * The CHECK.ABANDON subroutine calls CHECK.CHANGED. If any changes are
0619 * found, a message is displayed and the user is prompted to abandon the
0620 * changes. If there are no changes, or if the user decides to abandon
0621 * changes, the OK indicator variable is set to TRUE. Otherwise it is set
0622 * to FALSE.
0623 *
0624 CHECK.ABANDON: *
0625 *
0626 GOSUB CHECK.CHANGED
0627 IF NOT(OK) THEN
0628   PRINT EL:UNDREV:'Do you want to abandon all your changes? ':NORMAL:BEL:
0629   INPUT ANS:
0630   IF ANS[1,1] EQ 'Y' OR ANS[1,1] EQ 'y' THEN OK = 1
0631   PRINT EL:
0632 END
0633 RETURN
0634 *
0635 *
0636 *****
0637 *
0638 * The DSPSCRN subroutine is used to refresh the entire screen.
0639 *
0640 DSPSCRN: *
0641 *
0642 * Clear the screen
0643 PRINT CLR:NORMAL:
0644 *
0645 * Display heading & labels
0646 PRINT @(5,0):'Customer File Maintenance':
0647 FOR XLINE = 1 TO NUMFLDS
0648   LBWD = CONTROL(XLINE)<1,4>
0649   LBTX = (XLINE 'L#2 ') : CONTROL(XLINE)<1,1> ;* Prepend line number to label
0650   IF LBWD > 0 THEN LBTX = (LBTX : STR('.',LBWD))[1,LBWD] ;* Pad label with dots
0651   PRINT @(CONTROL(XLINE)<1,2>,CONTROL(XLINE)<1,3>):LBTX:
0652 NEXT XLINE
0653 *
0654 * Display the field data
0655 GOSUB DSPDATA
0656 RETURN
0657 *
0658 *
0659 *****
0660 *
0661 * The DSPDATA subroutine is used to refresh the field data for all fields.
0662 *
0663 DSPDATA: *
0664 *
0665 FOR XLINE = 1 TO NUMFLDS
0666   GOSUB DSPLINE
0667 NEXT XLINE
0668 RETURN
0669 *
```

```
0670 *
0671 *****
0672 *
0673 * The DSPLINE subroutine is used to refresh the field data for one field.
0674 * The field to be refreshed is specified by the XLINE variable. If the
0675 * field is active (XLINE = ACTIVE), then the field data is displayed using
0676 * the REVERSE display attribute. Otherwise it is displayed using the
0677 * DIMREV display attribute.
0678 *
0679 DSPLINE: *
0680 *
0681 MSK = 'L#':CONTROL(XLINE)<1,7>
0682 PRINT @(CONTROL(XLINE)<1,5>,CONTROL(XLINE)<1,6>):
0683 IF XLINE EQ ACTIVE THEN
0684     PRINT REVERSE:
0685 END ELSE
0686     PRINT DIMREV:
0687 END
0688 PRINT IDATA(XLINE) MSK:
0689 PRINT NORMAL:
0690 RETURN
0691 *
0692 END
0693
```