```
0001  *
0002  ************************************************************************
0003  * Copyright (c) 2004 Zumasys, Inc. as an unpublished work.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 4
0022  *
0023  ************************************************************************
0024  ************************************************************************
0025  *
0026  * This example illustrates a character-based interface using the Zumasys
0027  * Smart User Interface library for data entry. Most display and input
0028  * processing is handled by the SUI subroutines, rather than using PRINT
0029  * and INPUT statements. By using the SUI library, input fields accept
0030  * cursor keys, editing keys, function keys and mouse clicks. The SUI
0031  * subroutines also utilize AccuTerm Visual Styles to display screen
0032  * elements using a Windows-like look (if avaliable).
0033  *
0034  ************************************************************************
0035  *
0036  * To use the SUI library, you must include SUI.TERMDEF. This INCLUDE item
0037  * contains EQUates for various terminal functions like visual attributes
0038  * and line graphics, as well as constants for the keyboard commands like
0039  * TERM.ENTER$ or TERM.PGUP$.
0040  *
0041  $INCLUDE SUIBP SUI.TERMDEF
0042  *
0043  ************************************************************************
0044  *
0045  * The CUST.REC INCLUDE item declares the CUST.REC array, which is used to
0046  * store a working copy of the current data record. It also defines the
0047  * record layout by equating field names to array positions in the CUST.REC
0048  * array.
0049  *
0050  $INCLUDE UIEX.CUST.REC
0051  *
0052  ************************************************************************
0053  *
0054  * EQUates for control characters and delimiters and other constants
0055  *
0056  EQU VM TO CHAR(253)
0057  EQU BEL TO CHAR(7)
0058  EQU ESC TO CHAR(27)
0059  EQU STX TO CHAR(2)
0060  EQU CR TO CHAR(13)
0061  EQU FALSE TO 0
```

```
0062  EQU TRUE TO 1
0063  *
0064  ************************************************************************
0065  *
0066  * Any routine that uses the SUI library must call SUI.GET.TERM to
0067  * initialize the TermDef array. The TermDef array is a required argument
0068  * for all of the SUI subroutines. It is much better to call this routine
0069  * one time and pass the TermDef array between various routines instead of
0070  * calling it in each application subroutine that uses SUI.
0071  *
0072  CALL SUI.GET.TERM(MAT TermDef)
0073  *
0074  ************************************************************************
0075  *
0076  * Once the TermDef array is initialized by SUI.GET.TERM, the TERM.RESET
0077  * string is sent to the terminal to initialize the terminal state. The
0078  * reset string programs necessary function keys and initalizes any other
0079  * terminal state as required for each particular terminal type.
0080  *
0081  PRINT TERM.RESET:
0082  *
0083  ************************************************************************
0084  *
0085  * This program uses AccuTerm Imaging to display pictures associated with
0086  * records in the customer file. This EQUate defines the path where the
0087  * image files are located.
0088  *
0089  $INCLUDE UIEX.PIX.PATH
0090  *
0091  * Check for image support (no images if dumb terminal or AccuTerm Lite)
0092  *
0093  IF INDEX(TERM.SPECIAL,'I',1) THEN IMGFLG = TRUE ELSE IMGFLG = FALSE
0094  *
0095  ************************************************************************
0096  *
0097  * Open our files...
0098  *
0099  OPEN 'CUST.SAMPLE' TO FN.CUST ELSE PRINT 'NO CUST.SAMPLE FILE'; STOP
0100  OPEN 'CUST.SAMPLE.CTRL' TO FN.CUST.CTRL ELSE PRINT 'NO CUST.SAMPLE.CTRL FILE'; STOP
0101  OPEN 'CUST.SAMPLE.XREF' TO FN.CUST.XREF ELSE PRINT 'NO CUST.SAMPLE.XREF FILE'; STOP
0102  *
0103  ************************************************************************
0104  *
0105  * Define the data field control structures. NUMFLDS is the number of
0106  * fields. The CONTROL array defines the field control elements such as
0107  * field label, label position, field position and size, and field prompt.
0108  * The IDATA array stores the current field values, and is initialized
0109  * from the CUST.REC array when a data record is read from the file. When
0110  * the record is updated, values are copied from the IDATA array to the
0111  * CUST.REC array and the CUST.REC array is passed to the CUST.UPDATE
0112  * subroutine to update the data and index files (a separate subroutine is
0113  * used to update the file since the update process may handle other
0114  * functions like updating indexes).
0115  *
0116  NUMFLDS = 12
0117  DIM CONTROL(12)
0118  DIM IDATA(12)
0119  *
0120  * Define field control elements
0121  *  value 1: field label text
0122  *  value 2: field label column
```

```
0123  *  value 3: field label row
0124  *  value 4: field label width (0 for actual width, no padding)
0125  *  value 5: field data column
0126  *  value 6: field data row
0127  *  value 7: field data width
0128  *  value 8: field data height
0129  *  value 9: field prompt message
0130  CONTROL(1) = 'Customer ID???0?8??0?]Enter the ID, or name to search for, or N for
      next number'
0131  CONTROL(2) = 'Contact???0?8??0?]Enter the contact name'
0132  CONTROL(3) = 'Company name???0?8??0?]Enter the company name'
0133  CONTROL(4) = 'Address line 1???0?8??0?]Enter address line 1'
0134  CONTROL(5) = 'Address line 2???0?8??0?]Enter address line 2'
0135  CONTROL(6) = 'City???0?8??5?]Enter the city'
0136  CONTROL(7) = 'State or province???0?8??5?]Enter the state or province abbreviation'
0137  CONTROL(8) = 'Zip/postal code??0?0?8?0?0?]Enter the zip or postal code'
0138  CONTROL(9) = 'Country??1?0?8?1?8?]Enter the country'
0139  CONTROL(10) = 'Phone???2?0?8?2?5?]Enter the phone number'
0140  CONTROL(11) = 'Fax???3?0?8?3?5?]Enter the fax number'
0141  CONTROL(12) = 'Notes??4?0?8?4?0?]Enter any notes about this customer'
0142  *
0143  ************************************************************************
0144  *
0145  * Define some screen control strings for prompts & errors
0146  *
0147  PROMPT ''
0148  PL = @(5,22):TERM.CEOL
0149  EL = @(5,23):TERM.CEOL
0150  *
0151  ************************************************************************
0152  *
0153  * This program processes one customer record at a time, and is organized
0154  * using three nested loops. The outermost loop (the RECORD loop) is
0155  * executed once for each record accessed. The loop repeats until the XIT
0156  * control variable is set to TRUE.
0157  *
0158  XIT = FALSE
0159  LOOP UNTIL XIT DO
0160   *
0161   ***********************************************************************
0162   *
0163   * Before prompting for the customer ID, reset the internal data array
0164   * (IDATA) and CUST.ID variables, then clear the screen and display the
0165   * heading and field labels.
0166   *
0167   GOSUB RESTART
0168   GOSUB DSPSCRN
0169   *
0170   ***********************************************************************
0171   *
0172   * The middle loop is the ACTION loop. It is executed for the current
0173   * record until the user performs an action that terminates processing of
0174   * that record, such as exiting, cancelling, saving or deleting the
0175   * record. The field number to begin prompting (NXTFLD) is initialized to
0176   * 1, causing the ID field to be prompted first. The loop immediately
0177   * enters the PROMPT loop, followed by the field modification prompt. The
0178   * loop repeats until the DONE control variable is set to TRUE.
0179   *
0180   NXTFLD = 1
0181   DONE = FALSE
0182   LOOP
```

```
0183    *
0184    ************************************************************************
0185    *
0186    * The inner loop is the PROMPT loop. It is executed for each prompt
0187    * field as specified by the NXTFLD variable. The prompt loop simply
0188    * calls the local INPUT.FIELD subroutine which performs field input,
0189    * data validation and keyboard command decoding. The FIELD loop repeats
0190    * until the field number is greater than the number of fields, or until
0191    * the DONE control variable is set to TRUE. The DONE control variable
0192    * may be set to TRUE in the INPUT.FIELD subroutine, if the user enters a
0193    * NULL to for the item-ID.
0194    *
0195    LOOP
0196     CURFLD = NXTFLD
0197    UNTIL DONE OR CURFLD > NUMFLDS DO
0198     *
0199     ************************************************************************
0200     *
0201     * Prompt for the next field. Update the NXTFLD variable with the field
0202     * number to prompt next, taking into account cursor keys and mouse
0203     * clicks.
0204     *
0205     GOSUB INPUT.FIELD
0206    REPEAT ;* end of PROMPT loop
0207    *
0208    ************************************************************************
0209    *
0210    * If the FIELD loop exited with the DONE control variable set to TRUE,
0211    * bypass the modification prompt because no action is required.
0212    * Otherwise, prompt for which field to modify, or other actions such as
0213    * save or delete.
0214    *
0215    IF NOT(DONE) THEN
0216     *
0217     NXTFLD = NUMFLDS + 1 ;* assume we need to reprompt for action
0218     *
0219     PRINT PL:'Enter field number to modify or FI to save, DE to delete, EX to exit: ':
0220     INPUT ANS:
0221     PRINT EL:
0222     *
0223     ************************************************************************
0224     *
0225     * Decode the response
0226     *
0227     ANS = OCONV(ANS, 'MCU')
0228     BEGIN CASE
0229      CASE NUM(ANS) AND ANS >= 1 AND ANS <= NUMFLDS; NXTFLD = ANS
0230      CASE ANS EQ 'FI'; GOSUB SAVE.RECORD
0231      CASE ANS EQ 'DE'; GOSUB DELETE.RECORD
0232      CASE ANS EQ 'EX' OR ANS EQ ''; GOSUB CHECK.EXIT
0233     END CASE
0234     *
0235    END
0236    *
0237   UNTIL DONE DO REPEAT ;* end of ACTION loop
0238   *
0239  REPEAT ;* end of RECORD loop
0240  *
0241  ************************************************************************
0242  *
0243  * All done - clear the screen, restore the cursor and exit!
```

```
0244  PRINT TERM.CLEAR:TERM.CSRON:
0245  STOP
0246  *
0247  *
0248  ************************************************************************
0249  ************************************************************************
0250  * LOCAL SUBROUTINES
0251  ************************************************************************
0252  ************************************************************************
0253  *
0254  *
0255  ************************************************************************
0256  *
0257  * The INPUT.FIELD subroutine is the main prompting routine. This routine
0258  * displays the prompt string (from the CONTROL array), and enters a loop,
0259  * prompting for a specified field (CURFLD) and setting the next field
0260  * variable (NXTFLD) appropriately. The loop repeats until the NXTFLD
0261  * (initially set to CURFLD) changes. This causes the field prompt to be
0262  * repeated in case unrecognized keys are pressed (for example, the PGUP
0263  * key), or invalid data is entered (illegal customer ID, etc.) If a NULL
0264  * is entered for the ID field, and there is no current record, the ACTION
0265  * loop control variable, DONE, and the RECORD loop control variable, XIT,
0266  * are set to TRUE, and this routine exits. A mouse click is handled by
0267  * setting the NXTFLD variable to the field number that was clicked.
0268  *
0269  INPUT.FIELD:
0270  *
0271  ************************************************************************
0272  *
0273  * Display the prompt message
0274  *
0275  CALL SUI.DISPLAY.LABEL(5, 22, 74, 'L', 0, CONTROL(CURFLD)<1,9>, CMD, MAT TermDef)
0276  *
0277  ************************************************************************
0278  *
0279  * Initialize current value, next field & character positions
0280  *
0281  PREVAL = IDATA(CURFLD) ;* Save previous field value to detect change in value
0282  NXTFLD = CURFLD ;* Assume field number not changed
0283  BEGPOS = 0 ;* Set initial text display character position
0284  CURPOS = 0 ;* Set initial text cursor character position
0285  *
0286  ************************************************************************
0287  *
0288  * Prompt for this field until NXTFLD variable is updated
0289  *
0290  LOOP
0291   *
0292   ***********************************************************************
0293   *
0294   * Prompt for input with full text editing functions
0295   *
0296   CALL SUI.INPUT.TEXT(CONTROL(CURFLD)<1,5>, CONTROL(CURFLD)<1,6>,
      CONTROL(CURFLD)<1,7>, CONTROL(CURFLD)<1,8>, '', 0, IDATA(CURFLD), BEGPOS, CURPOS,
      CMD, MAT TermDef)
0297   PRINT TERM.CSRON:EL: ;* Turn cursor back on & clear the error line
0298   *
0299   ***********************************************************************
0300   *
0301   * Decode returned CMD
0302   *
```

```
0303   BEGIN CASE
0304    *
0305    ***********************************************************************
0306    *
0307    * Check for ENTER, TAB, UP, DOWN or mouse click
0308    *
0309    CASE CMD EQ TERM.ENTER$ OR CMD EQ TERM.TAB$ OR CMD EQ TERM.DOWN$
0310     NXTFLD = CURFLD + 1; * Move to next field
0311    CASE CMD EQ TERM.STAB$ OR CMD EQ TERM.UP$
0312     IF CURFLD > 1 THEN NXTFLD = CURFLD - 1 ;* Move to previous field
0313    CASE CMD EQ TERM.LEFTBUTTON$
0314     GOSUB CHECK.MOUSE ;* Move to clicked-on field
0315     IF CURFLD EQ 1 AND NXTFLD NE CURFLD THEN
0316      IF IDATA(1) EQ '' THEN NXTFLD = CURFLD ;* Mouse click in invalid field - ignore
      click
0317     END
0318    *
0319   END CASE
0320   *
0321   ***********************************************************************
0322   *
0323   * Check for any special values (like 'END' or 'EXIT')
0324   *
0325   IF OCONV(IDATA(CURFLD),'MCU') EQ 'END' THEN
0326    IDATA(CURFLD) = PREVAL ;* Restore previous value
0327    GOSUB CHECK.ABANDON ;* Ensure OK to loose changes
0328    IF OK THEN
0329     *
0330     ********************************************************************
0331     *
0332     * No changes, or user OKs abandoning them, so we are outa here!
0333     *
0334     DONE = TRUE
0335     XIT = TRUE
0336     RETURN
0337     *
0338    END ELSE
0339     *
0340     ********************************************************************
0341     *
0342     * User does not want to abandon changes, so refresh previous value &
0343     * reprompt
0344     *
0345     XLINE = CURFLD ;* Field number to refresh
0346     GOSUB DSPLINE ;* Redisplay the previous value
0347     NXTFLD = CURFLD ;* Reprompt
0348     *
0349    END
0350   END
0351   *
0352   ***********************************************************************
0353   *
0354   * Peform field data validation if next field number changed
0355   *
0356   IF NXTFLD NE CURFLD THEN
0357    BEGIN CASE
0358     *
0359     CASE CURFLD EQ 1
0360      *
0361      ********************************************************************
0362      *
```

```
0363      * Validate the ID field. If NULL, quit. If changed, read new record.
0364      *
0365      IF CUST.ID EQ '' AND IDATA(CURFLD) EQ '' THEN
0366       DONE = TRUE
0367       XIT = TRUE
0368       RETURN
0369      END
0370      *
0371      IF IDATA(CURFLD) NE PREVAL THEN
0372       ID = IDATA(CURFLD)
0373       GOSUB CHECK.ID ;* Check the newly entered ID handling index lookups
0374       IF OK THEN
0375        *
0376        ********************************************************************
0377        *
0378        * New ID (or result of index lookup) is good!
0379        *
0380        IDATA(CURFLD) = ID ;* Update the current field value in case of index lookup
0381        *
0382       END ELSE
0383        *
0384        ********************************************************************
0385        *
0386        * New ID is invalid
0387        *
0388        IDATA(CURFLD) = PREVAL ;* Restore previous value
0389        XLINE = CURFLD ;* Field number to refresh
0390        GOSUB DSPLINE ;* Redisplay the previous value
0391        NXTFLD = CURFLD ;* Reprompt
0392        *
0393       END
0394      END
0395      *
0396    END CASE
0397   END
0398   *
0399  WHILE CURFLD EQ NXTFLD DO REPEAT
0400  *
0401  RETURN
0402  *
0403  *
0404  ************************************************************************
0405  *
0406  * The CHECK.EXIT subroutine checks if any field data has changed, and
0407  * prompts if the user wants to abandon changes. If no changes, or the
0408  * user decides to abandon the changes, the DONE and XIT loop control
0409  * variables are set to TRUE, causing all three loops to terminate, and the
0410  * program itself to exit.
0411  *
0412  CHECK.EXIT: *
0413  *
0414  GOSUB CHECK.ABANDON
0415  IF OK THEN
0416   DONE = TRUE
0417   XIT = TRUE
0418   GOSUB HIDEPIX
0419  END
0420  RETURN
0421  *
0422  *
0423  ************************************************************************
```

```
0424  *
0425  * The CHECK.ID subroutine validates a newly entered item-ID. If the
0426  * current record has unsaved changes, the user is prompted to abandon the
0427  * changes. If no changes, or the user abandons the changes, and the new ID
0428  * is not NULL, an attempt is made to read a record using the new ID. If
0429  * the read is not successful, the ID is assumed to be a search string, and
0430  * the search subroutine is called to select an ID based on the search
0431  * string. If a valid ID is returned (or if one was initially entered), the
0432  * new record data is displayed. If the new ID is null, or if the search
0433  * routine did not return a valid ID, a warning message is displayed and
0434  * the OK indicator variable is set to FALSE. Otherwise it is set to TRUE.
0435  *
0436  CHECK.ID: *
0437  *
0438  ************************************************************************
0439  *
0440  * Make sure we don't have any unsaved data before changing the ID
0441  *
0442  GOSUB CHECK.ABANDON
0443  IF NOT(OK) THEN RETURN ;* Reprompt for the ID
0444  IF ID EQ '' THEN
0445   OK = FALSE ;* NULL is not a valid ID!
0446  END ELSE
0447   *
0448   ***********************************************************************
0449   *
0450   * Check if user wants new ID
0451   *
0452   IF ID EQ 'N' OR ID EQ 'n' THEN
0453    * Get next sequential ID
0454    READVU ID FROM FN.CUST.CTRL,'NEXT',2 THEN
0455     WRITEV ID + 1 ON FN.CUST.CTRL,'NEXT',2
0456     GOSUB RESTART
0457     IDATA(1) = ID
0458    END ELSE
0459     PRINT EL:TERM.DRV:'Next item counter record not found!':TERM.NV:BEL:
0460     INPUT ANS:
0461     PRINT EL:
0462     OK = FALSE
0463     RETURN
0464    END
0465   END ELSE
0466    *
0467    ***********************************************************************
0468    *
0469    * Try to read the customer record from the entered ID
0470    *
0471    GOSUB HIDEPIX ;* Hide previous cust picture before reading new one
0472    GOSUB READ.RECORD
0473    IF NOT(OK) THEN
0474     *
0475     ***********************************************************************
0476     *
0477     * The attempt to read a record failed - assume the ID is a search string
0478     *
0479     CALL UIEX.GET.CUST.IDX(XID,ID,FN.CUST.XREF,FN.CUST)
0480     GOSUB DSPSCRN ;* Refresh the screen after index lookup
0481     *
0482     ***********************************************************************
0483     *
0484     * If the user did not select an item in the search routine, reprompt
```

```
0485     *
0486     IF XID EQ '' THEN RETURN
0487     *
0488     ************************************************************************
0489     *
0490     * Try to read the customer record from the selected ID
0491     *
0492     ID = XID
0493     GOSUB HIDEPIX ;* Hide previous cust picture before reading new one
0494     GOSUB READ.RECORD
0495     *
0496    END
0497   END
0498  END
0499  *
0500  ************************************************************************
0501  *
0502  * If success, display the new record, otherwise show warning message
0503  *
0504  IF OK THEN
0505   GOSUB DSPDATA ;* Display new record
0506  END ELSE
0507   PRINT EL:TERM.DRV:'Please enter a valid customer ID!':TERM.NV:BEL:
0508  END
0509  RETURN
0510  *
0511  *
0512  ************************************************************************
0513  *
0514  * The READ.RECORD subroutine reads a new customer record from the file and
0515  * initializes the internal field data array (IDATA) from the record array
0516  * (CUST.REC). If the record does not exist, the routine returns with the
0517  * OK indicator variable set to FALSE. Otherwise OK is set to TRUE, and the
0518  * CUST.ID variable is set to the new ID.
0519  *
0520  READ.RECORD: *
0521  *
0522  MATREAD CUST.REC FROM FN.CUST,ID THEN
0523   CUST.ID = ID
0524   IDATA(1) = CUST.ID
0525   IDATA(2) = CUST.CONTACT
0526   IDATA(3) = CUST.NAME
0527   IDATA(4) = CUST.ADDRESS1
0528   IDATA(5) = CUST.ADDRESS2
0529   IDATA(6) = CUST.CITY
0530   IDATA(7) = CUST.ST
0531   IDATA(8) = CUST.ZIP
0532   IDATA(9) = CUST.COUNTRY
0533   IDATA(10) = CUST.PHONE
0534   IDATA(11) = CUST.FAX
0535   IDATA(12) = CUST.HISTORY
0536   OK = TRUE ;* Set the SUCCESS indicator
0537  END ELSE
0538   OK = FALSE ;* Set the FAILURE indicator
0539  END
0540  RETURN
0541  *
0542  *
0543  ************************************************************************
0544  *
0545  * The DELETE.RECORD subroutine confirms that the user intends to delete
```

```
0546  * the current record. If the action is confirmed, the CUST.DELETE
0547  * subroutine is called to perform the deletion. A separate subroutine is
0548  * used to handle updating indexes, etc.
0549  *
0550  DELETE.RECORD: *
0551  *
0552  IF CUST.ID NE '' THEN
0553   PRINT EL:TERM.DRV:'Are you sure you want to delete this customer? ':TERM.NV:
0554   INPUT ANS:
0555   IF ANS[1,1] EQ 'Y' OR ANS[1,1] EQ 'y' THEN
0556    * Deletion has been confirmed - do the delete
0557    OK = TRUE ;* Set the SUCCESS indicator
0558    CALL UIEX.CUST.DELETE(CUST.ID,FN.CUST,FN.CUST.XREF)
0559    DONE = TRUE ;* proceed to next record
0560    GOSUB HIDEPIX ;* erase picture
0561   END ELSE
0562    OK = FALSE ;* Set the FAILURE indicator
0563   END
0564  END
0565  RETURN
0566  *
0567  *
0568  ************************************************************************
0569  *
0570  * The SAVE.RECORD subroutine copies internal field data from the IDATA
0571  * array to the customer record array (CUST.REC). The CUST.UPDATE
0572  * subroutine is called to perform the update. A separate subroutine is
0573  * used to handle updating indexes, etc.
0574  *
0575  SAVE.RECORD: *
0576  *
0577  IF IDATA(1) NE '' THEN
0578   * Copy data from the internal field data array (IDATA) to the CUST.REC array
0579   CUST.ID = IDATA(1)
0580   CUST.CONTACT = IDATA(2)
0581   CUST.NAME = IDATA(3)
0582   CUST.ADDRESS1 = IDATA(4)
0583   CUST.ADDRESS2 = IDATA(5)
0584   CUST.CITY = IDATA(6)
0585   CUST.ST = IDATA(7)
0586   CUST.ZIP = IDATA(8)
0587   CUST.COUNTRY = IDATA(9)
0588   CUST.PHONE = IDATA(10)
0589   CUST.FAX = IDATA(11)
0590   CUST.HISTORY = IDATA(12)
0591   * Update the file
0592   CALL UIEX.CUST.UPDATE(CUST.ID,MAT CUST.REC,FN.CUST,FN.CUST.XREF)
0593   OK = TRUE ;* Set the SUCCESS indicator
0594  END ELSE
0595   OK = FALSE ;* Set the FAILURE indicator
0596  END
0597  DONE = TRUE ;* proceed to next record
0598  RETURN
0599  *
0600  *
0601  ************************************************************************
0602  *
0603  * The RESTART subroutine prepares the internal field data array (IDATA),
0604  * customer record array (CUST.REC) and ID for a new customer record.
0605  *
0606  RESTART: *
```

```
0607  *
0608  CUST.ID = ''
0609  MAT CUST.REC = ''
0610  MAT IDATA = ''
0611  RETURN
0612  *
0613  *
0614  ************************************************************************
0615  *
0616  * The CHECK.CHANGED subroutine checks if any internal field data has been
0617  * changed. The OK indicator variable is set to FALSE if any data is
0618  * changed, otherwise it is set to TRUE. The ID field is not checked, since
0619  * it is appropriately handled by the CHECK.ID subroutine.
0620  *
0621  CHECK.CHANGED: *
0622  *
0623  OK = FALSE
0624  IF CUST.CONTACT # IDATA(2) THEN RETURN
0625  IF CUST.NAME # IDATA(3) THEN RETURN
0626  IF CUST.ADDRESS1 # IDATA(4) THEN RETURN
0627  IF CUST.ADDRESS2 # IDATA(5) THEN RETURN
0628  IF CUST.CITY # IDATA(6) THEN RETURN
0629  IF CUST.ST # IDATA(7) THEN RETURN
0630  IF CUST.ZIP # IDATA(8) THEN RETURN
0631  IF CUST.COUNTRY # IDATA(9) THEN RETURN
0632  IF CUST.PHONE # IDATA(10) THEN RETURN
0633  IF CUST.FAX # IDATA(11) THEN RETURN
0634  IF CUST.HISTORY # IDATA(12) THEN RETURN
0635  OK = TRUE
0636  RETURN
0637  *
0638  *
0639  ************************************************************************
0640  *
0641  * The CHECK.ABANDON subroutine calls CHECK.CHANGED. If any changes are
0642  * found, a message is displayed and the user is prompted to abandon the
0643  * changes. If there are no changes, or if the user decides to abandon
0644  * changes, the OK indicator variable is set to TRUE. Otherwise it is set
0645  * to FALSE.
0646  *
0647  CHECK.ABANDON: *
0648  *
0649  GOSUB CHECK.CHANGED
0650  IF NOT(OK) THEN
0651   PRINT EL:TERM.DRV:'Do you want to abandon all your changes? ':TERM.NV:BEL:
0652   INPUT ANS:
0653   IF ANS[1,1] EQ 'Y' OR ANS[1,1] EQ 'y' THEN OK = 1
0654   PRINT EL:
0655  END
0656  RETURN
0657  *
0658  *
0659  ************************************************************************
0660  *
0661  * The DSPSCRN subroutine is used to refresh the entire screen.
0662  *
0663  DSPSCRN: *
0664  *
0665  * Clear the screen
0666  PRINT TERM.CLEAR:
0667  *
```

```
0668  * If running AccuTerm, display the logo
0669  IF IMGFLG THEN
0670   PRINT ESC:STX:'iL,':PIX.PATH:'ASE-LOGO-SMALL.JPG,65,0,12,2,0,N':CR:
0671  END
0672  *
0673  * Display heading & labels
0674  CALL SUI.DISPLAY.LABEL(5, 0, 0, 'L', 0, 'Customer File Maintenance', CMD, MAT
      TermDef)
0675  FOR XLINE = 1 TO NUMFLDS
0676   LBWD = CONTROL(XLINE)<1,4>
0677   LBTX = (XLINE 'L#2 ') : CONTROL(XLINE)<1,1> ;* Prepend line number to label
0678   IF LBWD > 0 THEN LBTX = (LBTX : STR('.',LBWD))[1,LBWD] ;* Pad label with dots
0679   CALL SUI.DISPLAY.LABEL(CONTROL(XLINE)<1,2>, CONTROL(XLINE)<1,3>, LBWD, 'L', 0,
      LBTX, CMD, MAT TermDef)
0680  NEXT XLINE
0681  *
0682  * Display the field data
0683  GOSUB DSPDATA
0684  RETURN
0685  *
0686  *
0687  ************************************************************************
0688  *
0689  * The DSPDATA subroutine is used to refresh the field data for all fields.
0690  *
0691  DSPDATA: *
0692  *
0693  FOR XLINE = 1 TO NUMFLDS
0694   GOSUB DSPLINE
0695  NEXT XLINE
0696  GOSUB SHOWPIX
0697  RETURN
0698  *
0699  *
0700  ************************************************************************
0701  *
0702  * The DSPLINE subroutine is used to refresh the field data for one field.
0703  * The field to be refreshed is specified by the XLINE variable.
0704  *
0705  DSPLINE: *
0706  *
0707  CALL SUI.DISPLAY.TEXT(CONTROL(XLINE)<1,5>, CONTROL(XLINE)<1,6>, CONTROL(XLINE)<1,7>,
      CONTROL(XLINE)<1,8>, 0, '', IDATA(XLINE), 0, CMD, MAT TermDef)
0708  RETURN
0709  *
0710  *
0711  ************************************************************************
0712  *
0713  * If running AccuTerm and an customer ID has been entered, try to display
0714  * the customer's picture for the current record.
0715  *
0716  SHOWPIX: *
0717  *
0718  IF IMGFLG THEN
0719   IF CUST.ID NE '' THEN
0720    PRINT ESC:STX:'iL,':PIX.PATH:'CUST\':CUST.ID:'.JPG,60,3,15,8,1,B':CR:
0721   END
0722  END
0723  RETURN
0724  *
0725  *
```

```
0726  ************************************************************************
0727  *
0728  * Erase the current customer picture before calling the lookup screen
0729  * because pictures always display over text.
0730  *
0731  HIDEPIX: *
0732  *
0733  IF IMGFLG THEN
0734   IF CUST.ID NE '' THEN
0735    PRINT ESC:STX:'iD,':PIX.PATH:'CUST\':CUST.ID:'.JPG':CR:
0736   END
0737  END
0738  RETURN
0739  *
0740  *
0741  ************************************************************************
0742  *
0743  * The CHECK.MOUSE subroutine is called if any SUI input routine returns
0744  * with CMD set to TERM.LEFTBUTTON$. This routine checks each field,
0745  * passing the field position and size to SUI.MOUSE.HIT. If a "hit" is
0746  * detected, the NXTFLD variable is set to the field number of the clicked
0747  * field. If no field was clicked, the position and size of the command bar
0748  * is checked, and if "hit", NXTFLG is set to NUMFLDS + 1 to cause the
0749  * ACTION and PROMPT loops to prompt for action using the command bar. The
0750  * actual mouse click location is saved internally by the SUI input
0751  * routines, and if a command button is clicked while a different field is
0752  * active, the click action is executed when SUI.INPUT.BTNBAR is called.
0753  *
0754  CHECK.MOUSE: *
0755  *
0756  FOR XLINE = 1 TO NUMFLDS
0757   CALL SUI.MOUSE.HIT(CONTROL(XLINE)<1,5>, CONTROL(XLINE)<1,6>, CONTROL(XLINE)<1,7>,
      CONTROL(XLINE)<1,8>, 0, HIT, MAT TermDef)
0758   IF HIT THEN
0759    NXTFLD = XLINE
0760    RETURN
0761   END
0762  NEXT XLINE
0763  RETURN
0764  *
0765  END
0766
```