Examples

This document contains the example programs and data areas listed in the following table. Source code is provided during the installation procedure and is located in the library SYSSPT.

See Examples for stored procedure link routines STPLNK01, STPLNK02, and STPLNK03.

Name	Description
SAMPINT1	This program issues a command that results in a trigger being fired and performs additional processing whenever an update or store record is written for a specific file.
SAMPPRC1	A stored procedure version of SAMPINT1. Data is passed as a parameter and information extracted from this parameter is returned.
SAMPP001	A stored procedure invoked from SAMPPRC1. It uses information from the input parameter to extract and create additional information to be returned to the caller.
STPLCB	An Adabas ACB control block layout used as a local data area (LDA) in example routines such as STPLNK02 and STPLNK03.
STPLCBE	An Adabas ACBX control block layout used as a local data area (LDA)
STPLRBE	An example layout of the parameter data area (PDA) that must be used when the record buffer extraction routine (STPRBE) is called.
STPUTPRM	An example of the parameter data area passed to the subprogram STPUTRAK.
STPUTRAK	The routine invoked by the Natural trigger driver when a request is received to execute a procedure. The routine is called only if the log trigger activity option in the Trigger Maintenance profile is set to ‘active’. The name of this routine must not be changed.
STPAPARM	An example layout of the parameter data area (PDA) that is passed from the Natural trigger driver when a procedure is invoked.
STPAPRM1	An example layout of the parameter data area (PDA) that is passed from the Natural trigger driver when a procedure is being invoked and STPUTRAK has requested that extended information also be passed. The extended area length is 250 bytes. It is maintained as part of the trigger driver’s global data area (GDA) and is kept intact across each invocation of a procedure by the trigger driver.
SAMP0001	The procedure that is invoked as the result of a trigger being fired for the commands originating from SAMPINT1.
SAMP0002	The procedure invoked for any command issued to the CONTACTS file, which is used as an example file in the SAMPINT1 program. The procedure audits all commands by writing a message to CMPRINT and creating a log record on the audit file.
SAMP0003	Like SAMP0002, this procedure audits commands issued to the CONTACTS file; however, it audits only participating commands.

Name	Description
SAMP0004	The procedure invoked through the execution of the SAMPREF1 routine for any deletes to the VEHICLES file. The example application is for referential logic type Restrict.
SAMP0005	The procedure invoked through the execution of the SAMPREF2 routine for any updates to the EMPLOYEES file. This is an example of referential integrity type Cascade.
SAMPREF1	This routine issues commands to invoke triggers that will perform some Referential integrity validation of the VEHICLES file. The associated procedure is SAMP0004, which performs Restrict checking on the records being deleted.
SAMPREF2	This routine issues commands to invoke triggers that will perform some Referential integrity checking for updates to the primary key on the EMPLOYEES file. The associated procedure is SAMP0005, which cascades any updates to the primary key to the foreign keys on the VEHICLES and MISCELLANEOUS files.

SAMPINT1

```
0010 ************************************************************************
0020 *  Application: Adabas Triggers
0030 *      Program: SAMPINT1
0040 *     Function: Routine to add names to file (CONTACTS). A trigger
0050 *               will be established to perform additional processing
0060 *               whenever the name is updated or added.
0070 * Trigger Defn: The definition on the trigger file (one for an update
0080 *               and one for the STORE/INSERT) is as follows:
0090 *                 File Number ....... 11
0100 *                 File Name ......... CONTACTS
0110 *                 Command Type ...... Update + Insert
0120 *                 Long Field Name ... CONTACT-NAME
0130 *                 Adabas Field ...... LE
0140 *                 Field Prty/Seq .... 10_
0150 *               Procedure Information
0160 *                 Name (Subpgm)...... SAMP0001
0170 *                 Pre Cmd Select .... Y  (Pre)
0180 *                 Trigger Type ...... P  (Participating)
0190 *                 CALLNAT Params .... C  (Cntl Info + Resp)
0200 *                 RecBuffer Access .. U  (May be Updated)
0210 *
0220 *               NOTE: An additional trigger also exists for this file
0230 *                     whereby any commands to the file will result in a
0240 *                     trigger being fired. This definition is:
0250 *
0260 *                 File Number ....... 11
0270 *                 File Name ......... CONTACTS
0280 *                 Command Type ...... ** All Command **
0290 *                 Long Field Name ... ** Any Field **
0300 *                 Adabas Field ...... **
0310 *                 Field Prty/Seq .... ___
0320 *               Procedure Information
0330 *                 Name (Subpgm)...... SAMP0002
0340 *                 Pre Cmd Select .... Y  (Pre)
0350 *                 Trigger Type ...... A  (Asynchronous)
```

```
0360 *                CALLNAT Params .... C  (Cntl Info + Resp)
0370 *                RecBuffer Access .. N  (No RecBuff Access)
0380 *
0390 *       Author: Adabas Development
0400 *         Date: December 1995
0410 ************************************************************************
0420 DEFINE DATA LOCAL
0430  01  #NAME     (A60)
0440  01  RESP       (N4)
0450  01  CONTACTS VIEW OF CONTACTS    /* file 11 for this example
0460    02  CONTACT-NAME               /* field LE,A,60,NU,DE
0470    02  CONTACT-UPPER              /* field LO,A,60,NU,DE
0480    02  KEYWORDS (20)              /* field LM,A,20,NU,MU
0490 END-DEFINE
0500 *
0510 REPEAT
0520 *
0530 INPUT (AD=WMIL'_' CD=NE)
0540      'Trigger Example for Data Consistancy' (YEI)
0550   // 'Name ...' (TU) #NAME
0560 *
0570 IF #NAME = MASK('.')              /* exit?
0580   STOP                            /* yes
0590 IF #NAME = ' '                    /* name must be specified
0600   REINPUT 'Invalid Name specified'
0610 *
0620 FIND CONTACTS WITH CONTACT-NAME = #NAME /* Find the name
0630   IF NO RECORDS FOUND             /* does it exist?
0640     DO                            /* no, so we should add it
0650 *
0660 *     Although only the CONTACT-NAME is being added, the other fields to
0670 *     be used in the Store are included. In this way, the format buffer
0680 *     and record buffer have reference to these files, since this example
0690 *     results in a pre-trigger that sets the values in these fields.
0700 *     Of course a post-trigger would also have worked; however, it would
0710 *     have been necessary for the procedure to do an additional read and
0720 *     update of the record. In this example, better performance is
0730 *     achieved with a pre-trigger.
0740 *
0750      RESET CONTACT-UPPER KEYWORDS(*)
0760      MOVE #NAME TO CONTACT-NAME    /* move value into view
0770 *
0780      STORE CONTACTS                /* insert the new record on the file
0790      END TRANSACTION               /* and commit the transaction
0800      IF *ISN(0780) = 0             /* we can check that a new ISN exists
0810        DO
0820          WRITE 'Store was unsuccessful' *ISN(0780)
0830          ESCAPE BOTTOM
0840        DOEND
0850      GET CONTACTS *ISN(0780)       /* now we refresh the record buffer
0860      INPUT (AD=O CD=TU)            /* and show the results of the Store
0870            '*** Results ***' (YEI)
0880          / 'Name ......' (GRI) CONTACT-NAME
0890          / 'Upper .....' (GR)  CONTACT-UPPER
0900          / 'Keywords ..' (GR) KEYWORDS (1:3)
0910          / '           ' KEYWORDS (4:6)
0920          / '           ' KEYWORDS (7:9)
0930          / '           ' KEYWORDS (10:12)
0940          / '           ' KEYWORDS (13:15)
0950          / '           ' KEYWORDS (16:18)
0960          / '           ' KEYWORDS (19:20)
0970      ESCAPE BOTTOM                 /* and exit the Add Record logic
```

```
0980    DOEND
0990  SET KEY PF3 PF5                 /* activate a couple of PF-keys
1000  INPUT (AD=O CD=TU)               /* allow the name to be changed
1010            '*** Make required Changes and PRESS PF5 to Update:' (YEI)
1020          / 'Name ......' (TU) CONTACT-NAME (AD=WMIL'_' CD=NE)
1030          / 'Upper .....' (TU) CONTACT-UPPER
1040          / 'Keywords ..' (TU) KEYWORDS (1:3)
1050          / '           ' KEYWORDS (4:6)
1060          / '           ' KEYWORDS (7:9)
1070          / '           ' KEYWORDS (10:12)
1080          / '           ' KEYWORDS (13:15)
1090          / '           ' KEYWORDS (16:18)
1100          / '           ' KEYWORDS (19:20)
1110    IF *PF-KEY = 'PF5'
1120      DO
1130        UPDATE(0620)               /* do the modification
1140        GET CONTACTS *ISN(0620)    /* now we refresh the record buffer
1150        INPUT (AD=O CD=TU)         /* and show the results of the update
1160              '*** Results of the Update ***' (YEI)
1170            / 'Name ......' (GRI) CONTACT-NAME
1180            / 'Upper .....' (GR)  CONTACT-UPPER
1190            / 'Keywords ..' (GR) KEYWORDS (1:3)
1200            / '           ' KEYWORDS (4:6)
1210            / '           ' KEYWORDS (7:9)
1220            / '           ' KEYWORDS (10:12)
1230            / '           ' KEYWORDS (13:15)
1240            / '           ' KEYWORDS (16:18)
1250            / '           ' KEYWORDS (19:20)
1260        MOVE CONTACT-NAME TO #NAME /* and reset the name
1270        ESCAPE BOTTOM              /* and exit the Update Record logic
1280      DOEND
1290  ESCAPE BOTTOM                   /* no update done
1300 CLOSE LOOP(0620)
1310 END TRANSACTION                  /* my job is to confirm and release
1320 CLOSE LOOP(0510)
1330 *
1340 END
```

SAMPPRC1

```
0010 ************************************************************************
0020 *  Application: Adabas Stored Procedures
0030 *      Program: SAMPPRC1
0040 *     Function: Routine to input a name and then invoke a stored
0050 *               procedure to populate additional fields based on the
0060 *               name passed.
0070 *
0080 *       Author: Adabas Development
0090 *         Date: December 1995
0100 ************************************************************************
0110 DEFINE DATA LOCAL
0120  01  #NAME     (A60)
0130  01  RESP       (N4)
0140  01  CONTACTS-INFORMATION         /* could be a file
0150    02  CONTACT-NAME  (A60)
0160    02  REDEFINE CONTACT-NAME
0170      03  PARM1     (A1/60)
0180    02  CONTACT-UPPER (A60)
0190    02  REDEFINE CONTACT-UPPER
0200      03  PARM2     (A1/60)
0210    02  KEYWORDS (A20/1:20)
```

```
0220    02  REDEFINE KEYWORDS
0230      03  PARM3    (A1/400)
0240  01  LINK-ROUTINE-PARMS          /* parameters for the link routine
0250    02  P-FUNC         (A1)
0260    02  P-PROC         (A8)       /* SAMP0001
0270    02  P-OPTIONS      (A8)       /* 'PCU'
0280    02  P-LEN        (P3/5)       /* lengths for the parameters
0290    02  P-MSG         (A72)       /* response message
0300    02  P-RESP         (N4)       /* response code
0310 END-DEFINE
0320 *
0330 MOVE '2'        TO P-FUNC         /* function....not relevant for this
0340 MOVE 'SAMPP001' TO P-PROC         /* procedure name
0350 MOVE 'NCU'      TO P-OPTIONS      /* non-partic + ctrl parms + upd RB
0360 MOVE 60         TO P-LEN(1) P-LEN(2)
0370 MOVE 200        TO P-LEN(3)
0380 MOVE 100        TO P-LEN(4) P-LEN(5)
0390 RESET P-MSG P-RESP
0400 *
0410 REPEAT
0420 *
0430 * In this example, the routine prompts the end user for an organization
0440 * name and in response, extracts some keywords from the value.
0450 * This is similar to SAMPINT1 (except that no file is being used) and
0460 * is a possible alternative to it.
0470 *
0480 INPUT (AD=WMIL'_' CD=NE)
0490 'Stored Procedure Example for Data Consistency' (YEI)
0500 // 'Name ...' (TU) #NAME
0510 *
0520 IF #NAME = MASK('.') /* exit?
0530 STOP /* yes
0540 IF #NAME = ' ' /* name must be specified
0550 REINPUT 'Invalid Name specified'
0560 *
0570 RESET CONTACT-UPPER KEYWORDS(*)
0580 MOVE #NAME TO CONTACT-NAME /* move value into view
0590 *
0600 CALLNAT 'STPLNK03' P-FUNC P-PROC P-OPTIONS P-LEN(1) PARM1(1:60)
0610 P-LEN(2) PARM2(1:60) P-LEN(3) PARM3(1:200)
0620 P-LEN(4) PARM3(201:300) P-LEN(5) PARM3(301:400)
0630 P-MSG P-RESP
0640 *
0650 INPUT (AD=O CD=TU) /* and show the results of the Store
0660 '*** Results ***' (YEI)
0670 / 'Name ......' (GRI) CONTACT-NAME
0680 / 'Upper .....' (GR) CONTACT-UPPER
0690 / 'Keywords ..' (GR) KEYWORDS (1:3)
0700 / ' ' KEYWORDS (4:6)
0710 / ' ' KEYWORDS (7:9)
0720 / ' ' KEYWORDS (10:12)
0730 / ' ' KEYWORDS (13:15)
0740 / ' ' KEYWORDS (16:18)
0750 / ' ' KEYWORDS (19:20)
0760 *
0770 CLOSE LOOP(0410)
0780 *
0790 END
```

SAMPP001

```
0010 ************************************************************************
0020 *  Application: Adabas Stored Procedures
0030 *  Subprogram : SAMPP001
0040 *  Author     : Adabas Development
0050 *  Date       : August 1995
0060 *  Function   : Sample routine of processing by a procedure
0070 *  Remarks    : This routine converts a name into uppercase and extracts
0080 *               all the keywords associated with it. Once processing is
0090 *               completed, control is returned to the caller.
0100 *
0110 *               Parameter RESP must be set to zero if processing is
0120 *               successful.
0130 *
0140 *  Parameters : Name1        (A60)
0150 *               Name2        (A60)
0160 *               Keyword(A20/01:10)
0170 *               Keyword(A20/11:15)
0180 *               Keyword(A20/16:20)
0190 *
0200 *  Rec Buffer : The record buffer will be available for update via a
0210 *               CALL to the external routine STPRBE.
0220 *
0230 ************************************************************************
0240 DEFINE DATA PARAMETER USING STPAPARM
0250            LOCAL     USING STPLRBE      /* parms for the Call routine
0260            LOCAL
0270  01  REC-BUFFER(A20/1:26)               /* max rec buffer passed to STPRBE
0280  01  REDEFINE REC-BUFFER                /* redefine this to get the def.
0290    02  INPUT-NAME   (A60)
0300    02  OUTPUT-NAME  (A60)
0310    02  KEYWORDS(A20/1:20)
0320  01  FUNC            (A4)
0330  01  SUB             (I2)
0340  01  SUB1            (I2)
0350  01  SUB2            (I2)
0360  01  W-UPPER        (A61)
0370  01  REDEFINE W-UPPER
0380    02  #UPPER       (A60)
0390    02  REDEFINE #UPPER
0400      03  CHAR   (A1/1:60)
0410  01  #KEYS     (A40/1:20)
0420 END-DEFINE
0430 *
0440 *  Option below is to audit any procedure activity.
0450 *
0460 * CALLNAT 'SAMP0002' REQ-AREA RESP
0470 *
0480 * Since the record buffer information is available to us, we can
0490 * now call the record buffer extraction routine (STPRBE) to obtain
0500 * the contents of the buffer.
0510 *
0520 * Function 'GR' -- GET RB Value using RB offset + length
0530 *     This enables the caller to obtain information based on a
0540 *     certain location; hence, RBE-OFFSET specifies the start
0550 *     position, and RBE-LENGTH specifies the length.
0560 *
0570 MOVE 1   TO RBE-OFFSET               /* start at the beginning
0580 MOVE 520 TO RBE-LENGTH               /* for a max length of 520 bytes
0590 MOVE 'GR' TO FUNC
0600 CALL 'STPRBE' 'GR' RBE-AREA REC-BUFFER(1)
```

```
0610 IF RBE-RESP NE 0
0620   PRINT *PROGRAM 'received an error from the STPRBE routine. Error:'
0630        RBE-ERROR 'subcode' RBE-SUBCODE 'for func GR'
0640   MOVE RBE-RESP TO RESP
0650   ESCAPE ROUTINE
0660 END-IF
0670 * PERFORM PRINT-REC-BUFFER           /* option to print the parms
0680 *
0690 * Change all lowercase to UPPERcase
0700 *
0710 MOVE INPUT-NAME TO #UPPER
0720 *
0730 EXAMINE #UPPER AND TRANSLATE INTO UPPER CASE
0740 *
0750 MOVE #UPPER TO OUTPUT-NAME           /* save the uppercase name
0760 *
0770 FOR SUB 1 60                         /* loop to remove all special chars.
0780   IF CHAR(SUB) = MASK(S)
0790     MOVE ' ' TO CHAR(SUB)
0800     ESCAPE TOP
0810   END-IF
0820 END-FOR
0830 *
0840 * We are now ready to extract keywords from our name. This sample is
0850 * very basic and may be made as complex as required.
0860 * This routine assumes a max. length of 20 and a max. num. of 20 keywords
0870 *
0880 EXAMINE FULL W-UPPER FOR FULL ' A '   REPLACE ' '
0890 EXAMINE FULL W-UPPER FOR FULL ' AND ' REPLACE ' '
0900 EXAMINE FULL W-UPPER FOR FULL ' AS '  REPLACE ' '
0910 EXAMINE FULL W-UPPER FOR FULL ' AT '  REPLACE ' '
0920 EXAMINE FULL W-UPPER FOR FULL ' ARE ' REPLACE ' '
0930 EXAMINE FULL W-UPPER FOR FULL ' BE '  REPLACE ' '
0940 EXAMINE FULL W-UPPER FOR FULL ' DO '  REPLACE ' '
0950 EXAMINE FULL W-UPPER FOR FULL ' FOR ' REPLACE ' '
0960 EXAMINE FULL W-UPPER FOR FULL ' HERE ' REPLACE ' '
0970 EXAMINE FULL W-UPPER FOR FULL ' IF '  REPLACE ' '
0980 EXAMINE FULL W-UPPER FOR FULL ' IN '  REPLACE ' '
0990 EXAMINE FULL W-UPPER FOR FULL ' IS '  REPLACE ' '
1000 EXAMINE FULL W-UPPER FOR FULL ' IT '  REPLACE ' '
1010 EXAMINE FULL W-UPPER FOR FULL ' OF '  REPLACE ' '
1020 EXAMINE FULL W-UPPER FOR FULL ' ON '  REPLACE ' '
1030 EXAMINE FULL W-UPPER FOR FULL ' OR '  REPLACE ' '
1040 EXAMINE FULL W-UPPER FOR FULL ' TO '  REPLACE ' '
1050 EXAMINE FULL W-UPPER FOR FULL ' THE ' REPLACE ' '
1060 EXAMINE FULL W-UPPER FOR FULL ' TOO ' REPLACE ' '
1070 EXAMINE FULL W-UPPER FOR FULL ' WAS ' REPLACE ' '
1080 EXAMINE FULL W-UPPER FOR FULL ' WITH ' REPLACE ' '
1090 EXAMINE #UPPER FOR FULL ' ' REPLACE ',' /* put delimiters in the string
1100 *
1110 RESET KEYWORDS(*)
1120 STACK TOP DATA #UPPER                  /* now we will separate each word
1130 INPUT (AD=I IP=ON) #KEYS(01:03) / #KEYS(04:06) / #KEYS(07:09)
1140                  / #KEYS(10:12) /* #KEYS(13:15) / #KEYS(16:18)
1150                  / #KEYS(19:20)
1160 *
1170 MOVE 1 TO SUB2
1180 MOVE #KEYS(1) TO KEYWORDS(1)
1190 FOR SUB 2 20                           /* now we remove all duplicates
1200   FOR SUB1 1 SUB
1210     IF #KEYS(SUB) = KEYWORDS(SUB1)
1220       RESET #KEYS(SUB)
```

```
1230     END-IF
1240   END-FOR
1250   IF #KEYS(SUB) NE ' '
1260     ADD 1 TO SUB2
1270     MOVE #KEYS(SUB) TO KEYWORDS(SUB2)   /* and finally save the value
1280   END-IF
1290 END-FOR
1300 *
1310 * Function 'UR' -- Update RB value using RB offset + length
1320 *     This enables the caller to change information based on a
1330 *     certain location; hence, RBE-OFFSET specifies the start
1340 *     position and RBE-LENGTH specified the length.
1350 *
1360 * PERFORM PRINT-REC-BUFFER              /* print the final results
1370 MOVE 1   TO RBE-OFFSET                 /* start at the beginning
1380 MOVE 520 TO RBE-LENGTH               /* for a max. length of 520 bytes
1390 MOVE 'UR' TO FUNC                    /* req to update all changes
1400 CALL 'STPRBE' 'UR' RBE-AREA REC-BUFFER(1)
1410 IF RBE-RESP NE 0
1420   PRINT *PROGRAM 'received an error from the STPRBE routine. Error:'
1430        RBE-ERROR 'subcode' RBE-SUBCODE 'for func UR'
1440   MOVE RBE-RESP TO RESP
1450   ESCAPE ROUTINE
1460 END-IF
1470 *
1480 * Return to the caller: everything went okay
1490 *
1500 ESCAPE ROUTINE
1510 *
1520 DEFINE SUBROUTINE PRINT-REC-BUFFER
1530 *----------------------------------------------------------------------*
1540 *
1550 *  For testing purposes, display the information returned from STPRBE
1560 *  This routine assumes a maximum of three subsystems running.
1570 *
1580 *----------------------------------------------------------------------*
1590   DECIDE ON FIRST VALUE OF RQ-TASK
1600     VALUE '01'
1610       WRITE (1) NOTITLE NOHDR (AD=L CD=TU)
1620           '**** RECORD BUFFER EXTRACTION: Function' FUNC '****'
1630           'Stored Procedure RBE' *PROGRAM '****'
1640         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
1650         / '             ....' (TU) RBE-ADA-FIELD  RBE-FIELD-OCC
1660         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR '<<<<<'
1670         / '  Message .......' (TU) RBE-MSG(AL=60)
1680         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
1690         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * * '
1700     VALUE '02'
1710       WRITE (2) NOTITLE NOHDR (AD=L CD=TU)
1720           '**** RECORD BUFFER EXTRACTION: Function' FUNC '****'
1730           'Stored Procedure RBE' *PROGRAM '****'
1740         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
1750         / '             ....' (TU) RBE-ADA-FIELD  RBE-FIELD-OCC
1760         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR '<<<<<'
1770         / '  Message .......' (TU) RBE-MSG(AL=60)
1780         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
1790         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * * '
1800     VALUE '03'
1810       WRITE (3) NOTITLE NOHDR (AD=L CD=TU)
1820           '**** RECORD BUFFER EXTRACTION: Function' FUNC '****'
1830           'Stored Procedure RBE' *PROGRAM '****'
1840         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
```

```
1850         / '             ....' (TU) RBE-ADA-FIELD  RBE-FIELD-OCC
1860         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR '<<<<<'
1870         / '  Message .......' (TU) RBE-MSG(AL=60)
1880         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
1890         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * * '
1900     NONE
1910       WRITE NOTITLE NOHDR (AD=L CD=TU)
1920           '**** RECORD BUFFER EXTRACTION: Function' FUNC '****'
1930           'Stored Procedure RBE' *PROGRAM '****'
1940         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
1950         / '             ....' (TU) RBE-ADA-FIELD  RBE-FIELD-OCC
1960         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR '<<<<<'
1970         / '  Message .......' (TU) RBE-MSG(AL=60)
1980         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
1990         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * * '
2000   END-DECIDE
2010 *
2020 END-SUBROUTINE
2030 *
2040 END
```

STPLCB

```
0010 ************************************************************************
0020 **
0030 **Local data area  'STPLCB'
0040 **describes Adabas control block
0050 **
0060 ************************************************************************
0070 DEFINE DATA LOCAL
0080 1 CB         (B80)
0090 1 REDEFINE CB
0100 2 CB-DSECT             /* ACB definition
0110 3 CB-CALL-TYPE(B1)
0120 3 CB-HOST-ID  (B1)
0130 2 CB-CMD      (A2)     /* command code
0140 2 CB-CID      (A4)     /* command ID
0150 2 CB-FILE     (B2)         /* file number
0160 2 REDEFINE CB-FILE
0170 3 CB-DBID     (B1)         /* one-byte DBNR
0180 3 CB-FNR      (B1)         /* one-byte FNR
0190 2 CB-RSP      (B2)         /* response code
0200 2 CB-ISN(B4)               /* ISN value
0210 2 CB-ISLL(B4)              /* ISN lower limit
0220 2 CB-ISQ(B4)               /* ISN quantity
0230 2 CB-FBL(B2)               /* format buffer length
0240 2 CB-RBL(B2)               /* record buffer length
0250 2 CB-SBL(B2)               /* search buffer length
0260 2 CB-VBL(B2)               /* value buffer length
0270 2 CB-IBL(B2)               /* ISN buffer length
0280 2 CB-CO1(A1)               /* command option 1
0290 2 CB-CO2(A1)               /* command option 2
0300 2 CB-ADD1(A8)              /* additions 1
0310 2 CB-ADD2(A4)              /* additions 2
0320 2 CB-ADD3(A8)              /* additions 3
0330 2 CB-ADD4(A8)              /* additions 4
0340 2 CB-ADD5(A8)              /* additions 5, reserved
0350 2 CB-CT(B4)                /* command time
```

```
0360 2 CB-UA(B4)                /* user area
0370 ************************************************************************
0380 ******* END OF LOCAL DATA AREA ***************************************
0390 ************************************************************************
```

STPLCBE

```
0010   * * ****************************** * **** ******************************
0020   * *
0030   * * LOCAL DATA AREA  'STPLCBE'
0040   * * DESCRIBES ADABAS CONTROL BLOCK
0050   * * EXTENDED
0060   * *
0070   * * ****************************** * **** ******************************
0080     1 CB                             B  192
0090   R 1 CB
0100     2 CB-DSECT                                /* ACBE DEFINITION
0110     3 CB-TYPE                        B    1
0120     3 CB-HOST                        A    1
0130     2 CB-VERSION                     A    2 /* ACBE VERSION
0140     2 CB-LENGTH                      B    2 /* ACBE LENGTH
0150     2 CB-CMD                         A    2 /* COMMAND
0160     2 CB-NUCID                       B    2 /* NUCID
0170     2 CB-RSP                         B    2 /* RESPONSE CODE
0180     2 CB-CID                         B    4 /* COMMAND ID
0190     2 CB-DBID                        B    4 /* DBID
0200     2 CB-FNR                         B    4 /* FILE NUMBER
0210     2 CB-RESERVED-1                  B    4 /* UNUSED
0220     2 CB-ISN                         B    4 /* ISN VALUE
0230     2 CB-RESERVED-2                  B    4 /* UNUSED
0240     2 CB-ISLL                        B    4 /* ISN LOWER LIMIT
0250     2 CB-RESERVED-3                  B    4 /* UNUSED
0260     2 CB-ISQ                         B    4 /* ISN QUANTITY
0270     2 CB-CO1                         A    1 /* COMMAND OPTION 1
0280     2 CB-CO2                         A    1 /* COMMAND OPTION 2
0290     2 CB-CO3                         A    1 /* COMMAND OPTION 3
0300     2 CB-CO4                         A    1 /* COMMAND OPTION 4
0310     2 CB-CO5                         A    1 /* COMMAND OPTION 5
0320     2 CB-CO6                         A    1 /* COMMAND OPTION 6
0330     2 CB-CO7                         A    1 /* COMMAND OPTION 7
0340     2 CB-CO8                         A    1 /* COMMAND OPTION 8
0350     2 CB-ADD1                        A    8 /* ADDITIONS 1
0360     2 CB-ADD2                        B    4 /* ADDITIONS 2
0370     2 CB-ADD3                        A    8 /* ADDITIONS 3
0380     2 CB-ADD4                        A    8 /* ADDITIONS 4
0390     2 CB-ADD5                        B    8 /* ADDITIONS 5
0400     2 CB-ADD6                        A    8 /* ADDITIONS 6
0410     2 CB-RESERVED-4                  B    4 /* RESERVED
0420     2 CB-ERROR                       B   16 /* SUPPLEMENTAL ERROR INFO
0430   R 2 CB-ERROR
0440     3 CB-ERROR-G                     B    4 /* OFFSET IN BUFFER 64-BIT
0450     3 CB-ERROR-A                     B    4 /* OFFSET IN BUFFER 32-BIT
0460     3 CB-ERROR-B                     A    2 /* ERROR CHARACTER FIELD
0470     3 CB-ERROR-C                     B    2 /* SUBCODE
0480     3 CB-ERROR-D                     A    1 /* ERROR BUFFER ID
0490     3 CB-ERROR-E                     B    3 /* BUFFER SEQ. NUMBER
0500     2 CB-SUB                         B    8 /* SUBCOMPONENT ERROR INFO
0510   R 2 CB-SUB
0520     3 CB-SUB-R                       B    2 /* SUBCOMPONENT RSP CODE
0530     3 CB-SUB-S                       B    2 /* SUBCOMPONENT REASON CD
0540     3 CB-SUB-T                       A    4 /* SUBCOMPONENT ERROR TEXT
```

```
0550     2 CB-LCMP                          B    8 /* COMPRESSED RECORD LNGTH
0560     2 CB-LDEC                          B    8 /* DECOMPRESSED LENGTH
0570     2 CB-TIME                          B    8 /* COMAND TIME
0580     2 CB-USER                          B   16 /* USER FIELD
0590     2 CB-ROUTER                        B    1 /* ROUTER FLAGS
0600     2 CB-RESERVED-5                    B   23 /* RESERVED
0610   * * ********************************* * **** ********************************
0620   * * ***** END OF LOCAL DATA AREA *** * **** ********************************
0630   * * ********************************* * **** ********************************
```

STPLRBE

```
***** DEFINE DATA LOCAL
0010 1  RBE-AREA(A154)      /* record buffer extraction area
0020 1  REDEFINE RBE-AREA
0030 2  RBE-MSG(A72)            /* error text for errors
0040 2  RBE-RESP(B4)            /* error number
0050 2  REDEFINE RBE-RESP
0060 3  RBE-SUBCODE(B2)         /* error subcode
0070 3  RBE-ERROR(B2)           /* actual error code
0080 2  RBE-VERNO(A4)           /* structure version
0090 2  RBE-FIELD-NAME(A32)     /* long name of field
0100 2  RBE-FORMAT(A1)      /* field format
0110 2  RBE-OPTS(A3)            /* special options
0120 2  RBE-LENGTH(B4)      /* field/RB length
0130 2  RBE-ADA-FIELD(A2)       /* Adabas short name
0140 2  RBE-RESRV2(A2)      /* reserved
0150 2  RBE-FIELD-OCC(B4)       /* field occurrence for MU or PE
0160 2  RBE-GROUP-OCC(B4)       /* PE occurrence for MU within PE
0170 2  RBE-OFFSET(B4)      /* offset into RB
0180 2  RBE-UNUSED(A18)         /* not used
***** END-DEFINE
```

STPUTPRM

```
***** DEFINE DATA PARAMETER
0010 1  CALL-TYPE(A1)       /* type of call
0020 **                     /* 'B' before invoking
0030 **                     /* 'A' after  invoking
0040 **                     /* 'E' error  incurred
0050 1  REQ-AREA(A200)      /* request area
0060 1  REDEFINE REQ-AREA
0070 2  RQ-VERNO(A4)        /* structure version
0080 2  RQ-TASK(A2)         /* subsystem number
0090 2  RQ-PROC(A8)         /* procedure name
0100 2  RQ-USER(A32)        /* user identification
0110 2  RQ-CMD(A2)          /* trigger command
0120 2  RQ-DBID(B2)         /* Trigger DBID
0130 2  RQ-FNR(B2)          /* trigger target file number
0140 2  RQ-FIELD(A2)        /* trigger field (short name)
0150 2  RQ-SYNC(A1)         /* sync/async request
0160 2  RQ-PARTIC(A1)       /* participating/non-participating request
0170 2  RQ-LENGTH(B2)       /* record buffer length
0180 2  RQ-UPD(A1)          /* RB update indicator
0190 2  RQ-TTYP(A1)         /* trigger type "P"re or Po"S"t
0200 2  RQ-RESP(B4)
0210 2  REDEFINE RQ-RESP
0220 3  RESP-CODE(B2)
0230 3  SUB-CODE(B2)
0240 2  RQ-PDA-TYPE(A1)     /* calling type
```

```
0250 2  RQ-RESRVED2(A55)
0260 2  RQ-CB(A80)          /* trigger control block
0270 1  ERR-INFO(A72)       /* error information for CALL-TYPE 'E'
0280 1  REDEFINE ERR-INFO
0290 2  ERR-NR(N4)          /* error number
0300 2  ERR-LINE(N4)        /* line number of error
0310 2  ERR-STAT(A1)        /* error status indicator
0320 2  ERR-PROG(A8)        /* error program
0330 2  ERR-LEVEL(N2)       /* not used
0340 2  ERR-TYPE(A24)       /* error identification
0350 1  REQ-GLOBAL-WS(A250) /* global WS area
0360 1  RESP(B4)            /* procedure response
***** END-DEFINE
```

STPUTRAK

```
0010 ************************************************************************
0020 *  Application: Adabas Stored Procedures
0030 *  Program    : STPUTRAK
0040 *  Function   : Routine that is invoked if triggers is running with
0050 *               'Trigger Logging' set to Active.
0060 *               Invoked (CALL-TYPE):
0070 *                  'I' - when the subsystem is initialized
0080 *                  'T' - when the subsystem is being terminated
0090 *                  'B' - before a procedure is invoked
0100 *                  'A' - after a procedure has completed
0110 *                  'E' - whenever an error occurs
0120 *  NOTE       : If logging is active, then the module (cataloged object)
0130 *               must exist; otherwise, a NAT0082 occurs.
0140 *  Author     : Adabas Development
0150 *  Date       : June 1994
0160 ************************************************************************
0170 DEFINE DATA PARAMETER USING STPUTPRM
0180            LOCAL
0190 01  EVENT      (A14)              /* event criteria
0200 01  REDEFINE EVENT
0210   02  E-FNR     (N5)
0220   02  E-F1      (A2)
0230   02  E-CMD     (A2)
0240   02  E-F2      (A2)
0250   02  E-FIELD   (A2)
0260 01  PARM-TYPE   (A7)
0270 01  TRIG-TYPE  (A10)
0280 01  REDEFINE TRIG-TYPE
0290   02  PRE-POST  (A4)
0300   02  FILL      (A1)
0310   02  PROC-TYPE (A5)
0320 01  UQE-ID     (A28)
0330 01  RB-TYPE     (A6)
0340 01  RBLEN       (N5)
0350 01  RESP-BIN    (B4)
0360 01  REDEFINE RESP-BIN
0370   02  RESP-SUBC (B2)
0380   02  RESP-CDE  (B2)
0390 END-DEFINE
0400 *
0410 FORMAT PS=0 LS=133                /* set report attributes
0420 *
0430 IF CALL-TYPE = 'I'                /* subsystem initialization
0440   WRITE NOTITLE (CD=TU)
0450        '***** Triggers and Stored Procedures *****' (GRI)
```

```
0460     / '      - Natural Subsystem Initialization -' (YEI)
0470    // 'Program + Library Location ...' (TU) *PROGRAM *LIBRARY-ID
0480     / 'Task Initialization Time .....' (TU) *DATX *TIMX
0490     / 'Task Identification Number ...' (TU) RQ-TASK
0500     / 'Task User Identification .....' (TU) *INIT-USER *INIT-ID
0510     / '********************** INIT ********************' (GRI)
0520  NEWPAGE                            /* setup for proper headings
0530  ESCAPE ROUTINE                     /* return control
0540 END-IF
0550 *
0560 IF CALL-TYPE = 'T'                  /* subsystem termination
0570  WRITE NOTITLE (CD=TU)
0580       '***** Triggers and Stored Procedures *****' (GRI)
0590     / '        - Natural Subsystem Termination -' (YEI)
0600    // 'Task Termination Time ........' (TU) *DATX *TIMX
0610     / 'Task Identification Number ...' (TU) RQ-TASK
0620     / 'Task User Identification .....' (TU) *INIT-USER *INIT-ID
0630     / '********************** EXIT *********************' (GRI)
0640  ESCAPE ROUTINE                     /* return control
0650 END-IF
0660 *
0670 IF CALL-TYPE = 'A'                  /*  after invoking procedure
0680   MOVE RESP TO RESP-BIN
0690   WRITE RQ-TASK 2X *DATX *TIMX 'complete' 16X RQ-PROC 6X 'Resp:'
0700         RESP-CDE RESP-SUBC
0710 END-IF
0720 *
0730 IF CALL-TYPE = 'B'                  /*  before invoking procedure
0740  MOVE RQ-RESRVED2 TO UQE-ID
0750* MOVE RB-RBL   TO RBLEN
0760  MOVE RQ-LENGTH TO RBLEN
0770  IF RQ-TTYP = 'P'
0780    MOVE 'Pre ' TO PRE-POST
0790  ELSE
0800    MOVE 'Post' TO PRE-POST
0810  END-IF
0820  IF RQ-SYNC = 'A'
0830    MOVE 'ASync' TO PROC-TYPE
0840  END-IF
0850  IF RQ-SYNC = 'S'
0860    MOVE 'Sync' TO PROC-TYPE
0870    IF RQ-PARTIC = 'P'
0880      MOVE 'Part' TO PROC-TYPE
0890    END-IF
0900    IF RQ-PARTIC = 'N'
0910      MOVE 'Non-P' TO PROC-TYPE
0920    END-IF
0930  END-IF
0940  DECIDE ON FIRST VALUE OF RQ-PDA-TYPE
0950    VALUE 'C' MOVE 'Control' TO PARM-TYPE
0960    VALUE 'N' MOVE 'No Parm' TO PARM-TYPE
0970    VALUE 'R' MOVE 'Resp'    TO PARM-TYPE
0980    NONE      MOVE 'Unknown' TO PARM-TYPE
0990  END-DECIDE
1000  DECIDE ON FIRST VALUE OF RQ-UPD
1010    VALUE 'A' MOVE 'Access' TO RB-TYPE
1020    VALUE 'N' MOVE 'No Rec' TO RB-TYPE
1030    VALUE 'U' MOVE 'Update' TO RB-TYPE
1040    NONE      MOVE '??????' TO RB-TYPE
1050  END-DECIDE
1060  IF RQ-CMD = 'PC'
1070    MOVE '*Stored Proc.*' TO EVENT
```

```
1080  ELSE
1090    MOVE RQ-FNR TO E-FNR
1100    MOVE RQ-CMD TO E-CMD
1110    MOVE RQ-FIELD TO E-FIELD
1120  END-IF
1130  DISPLAY NOTITLE (CD=TU)
1140      'Tsk' RQ-TASK 'Date' *DATX 'Time' *TIMX 'User' RQ-USER(AL=8)
1150      'Fnr    Cmd Fld' (TU) EVENT 'Proc' (TU) RQ-PROC
1160      'Type' (TU) TRIG-TYPE 'Parms' (TU) PARM-TYPE
1170      'RecBuf' RB-TYPE
1180 * PRINT 5X 'UserID ...' UQE-ID(EM=H(28)) /* UQE-ID
1190 *
1200 * A special overwrite option allows the user to have the procedure
1210 * called with the additional parameter of the RQ-GLOBAL-WS.
1220 * This is valid only if RQ-PDA-TYPE is set to 'C'.
1230 * The procedure should expect parameters to be passed as specified in
1240 * STPAPRM1 i.e. CALLNAT 'procname' REQ-AREA REQ-GLOBAL-WS RESP
1250 *
1260 * MOVE 'W' TO CALL-TYPE
1270 *
1280 * RQ-GLOBAL-WS is an area that is not changed between each call to
1290 * the procedures. It may be used for keeping statistics or whatever.
1300 *
1310 END-IF
1320 *
1330 IF CALL-TYPE = 'E'                   /* subsystem error notification
1340   WRITE NOTITLE (CD=TU)
1350        '***** Triggers and Stored Procedures *****' (GRI)
1360      / '        - Natural Subsystem Error Info -' (YEI)
1370     // 'Task Termination Time .......' (TU) *DATX *TIMX
1380     / 'Task Identification Number ..' (TU) RQ-TASK
1390       '<<< Interrupted with an ERROR <<<<<'
1400    // '* Subsystem Error Number ....' ERR-NR
1410   41T '* Stored Proc ....' RQ-PROC
1420     / '*           Active Module ...' ERR-PROG
1430   41T '* UserID .........' RQ-USER
1440     / '*           Line Number .....' ERR-LINE (EM=9999)
1450   41T '* Trigger Cmd ....' RQ-CMD
1460     / '*           Error Level .....' ERR-LEVEL
1470   41T '* Trigger Fnr ....' RQ-FNR (AD=L)
1480     / '*           Error Status ....' ERR-STAT
1490   41T '* Trigger Type ...' RQ-TTYP RQ-PARTIC 'opt' RQ-PDA-TYPE RQ-UPD
1500     / '*           Error Type ......' ERR-TYPE (AL=14)
1510*  41T '* Field Name .....' RQ-TRG-FIELD '+' RB-RBL
1520*    / '*           UQE Ident. ......' CA-USER
1530     /  '*** Processing for this request ABNORMALLY terminated ***'
1540    // '********************* ERROR *********************' (GRI)
1550  ESCAPE ROUTINE                     /* return control
1560 END-IF
1570 *
1580 END
```

STPAPARM

```
***** DEFINE DATA PARAMETER
0010 1  REQ-AREA(A200)      /* request area
0020 1  REDEFINE REQ-AREA
0030 2  RQ-VERNO(A4)        /* structure version
0040 2  RQ-TASK(A2)         /* subsystem number
0050 2  RQ-PROC(A8)         /* procedure name
0060 2  RQ-USER(A32)        /* user identification
```

```
0070 2  RQ-CMD(A2)          /* trigger command
0080 2  RQ-DBID(B2)         /* Trigger DBID
0090 2  RQ-FNR(B2)          /* trigger target file number
0100 2  RQ-FIELD(A2)        /* trigger field (short name)
0110 2  RQ-SYNC(A1)         /* sync/async request
0120 2  RQ-PARTIC(A1)       /* participating/non-participating request
0130 2  RQ-LENGTH(B2)       /* record buffer length
0140 2  RQ-UPD(A1)          /* RB update indicator
0150 2  RQ-TTYPE(A1)        /* pre- or post-trigger
0160 2  RQ-RESP(B4)         /* subcode(B2) + resp code(B2)
0170 2  RQ-PDA-TYPE(A1)     /* calling parameters type
0180 2  RQ-USERID(A28)      /* user ID from Adabas CQE
0190 2  RQ-RESRVED2(A27)
0200 2  RQ-CB(A80)          /* trigger control block
0210 1  RESP(B4)            /* procedure response
***** END-DEFINE
```

STPAPRM1

```
***** DEFINE DATA PARAMETER STPAPRM1 LIBRARY SYSSPT
0010 1  REQ-AREA(A200)           /* Request area
0020 1  REDEFINE REQ-AREA
0030 2  RQ-VERNO(A4)             /* structure version
0040 2  RQ-TASK(A2)              /* subsystem number
0050 2  RQ-PROC(A8)              /* procedure name
0060 2  RQ-USER(A32)             /* user identification
0070 2  RQ-CMD(A2)               /* trigger command
0080 2  RQ-DBID(B2)              /* Trigger DBID
0090 2  RQ-FNR(B2)               /* trigger target file number
0100 2  RQ-FIELD(A2)             /* trigger field (short name)
0110 2  RQ-SYNC(A1)              /* sync/async request
0120 2  RQ-PARTIC(A1)            /* participating/non-participating request
0130 2  RQ-LENGTH(B2)            /* record buffer length
0140 2  RQ-UPD(A1)               /* RB update indicator
0150 2  RQ-RESRVED1(A1)          /* not used
0160 2  RQ-RESP(B4)
0170 2  RQ-PDA-TYPE(A1)
0180 2  RQ-RESRVED2(A55)
0190 2  RQ-CB(A80)               /* trigger control block
0200 1  REQ-GLOBAL-WS(A250)      /* global WS area
0210 1  RESP(B4)                 /* procedure response
***** END-DEFINE
```

STPXPARM

```
0010     1 REQ-AREA                    A  200 /* Request Area
0020   R 1 REQ-AREA                           /* Redef. begin : REQ-AREA
0030     2 RQ-VERNO                    A    4 /* Structure Version
0040     2 RQ-TASK                     A    2 /* Subsystem Number
0050     2 RQ-PROC                     A    8 /* Procedure Name
0060     2 RQ-USER                     A   32 /* User Identification
0070     2 RQ-CMD                      A    2 /* Trigger Cmd
0080     2 RQ-DBID                     B    2 /* Trigger DBID
0090     2 RQ-FNR                      B    2 /* Trigger Target File Nr
0100     2 RQ-FIELD                    A    2 /* Trigger Field (Short Name)
0110     2 RQ-SYNC                     A    1 /* Sync/Async Request
0120     2 RQ-PARTIC                   A    1 /* Part/Non-participating Req
0130     2 RQ-LENGTH                   B    2 /* Record Buffer Length
0140     2 RQ-UPD                      A    1 /* RB Update Indicator
0150     2 RQ-TTYPE                    A    1 /* Pre or Post Trigger
```

```
0160     2 RQ-RESP                          B    4 /* Subcode(B2) + Resp Code(B2)
0170     2 RQ-PDA-TYPE                      A    1 /* Calling Parameters Type
0180     2 RQ-USERID                        A   28 /* UserID from ADABAS UQE
0190     2 RQ-RESRVED2                      A   27 /*
0200     2 RQ-CB                            A   80 /* Trigger Control Block
0210     1 RQ-CBX                           A  192 /* X Verion of CB
0220     1 RESP                             B    4 /* Procedure Response
```

SAMP0001

```
0010 ************************************************************************
0020 *  Application: ADASTP
0030 *  Subprogram : SAMP0001
0040 *  Author     : Adabas Development
0050 *  Date       : August 1995
0060 *  Function   : Sample routine of processing by a procedure
0070 *  Remarks    : This routine converts the CONTACT-NAME into uppercase
0080 *               and extracts all the keywords associated with it.
0090 *               Once processing is completed, control is returned to
0100 *               the caller.
0110 *               Parameter RESP must be set to zero if processing is
0120 *               successful.
0130 *
0140 *  Parameters : REQ-AREA (A200)
0150 *               RESP       (B4)
0160 *
0170 *  Trigger Typ: The type of trigger will be PARTICIPATING; i.e.,
0180 *               synchronous.
0190 *  Rec Buffer : The record buffer will be available for update via a
0200 *               call to the external routine STPRBE.
0210 * Trigger Defn: Definition on the trigger file (note that there is a
0220 *               trigger for the insert and update) is as follows:
0230 *                 File Number ....... 11
0240 *                 File Name ......... CONTACTS
0250 *                 Command Type ...... Update + Insert
0260 *                 Long Field Name ... CONTACT-NAME
0270 *                 Adabas Field ...... LE
0280 *                 Field Prty/Seq .... 10_
0290 *               Procedure Information
0300 *                 Name (Subpgm)...... SAMP0001
0310 *                 Pre Cmd Select .... Y  (Pre)
0320 *                 Trigger Type ...... P  (Participating)
0330 *                 CALLNAT Params .... C  (Cntl Info + Resp)
0340 *                 RecBuffer Access .. U  (May be updated)
0350 *
0360 ************************************************************************
0370 DEFINE DATA PARAMETER USING STPAPARM
0380           LOCAL     USING STPLRBE /* parms for the call routine
0390           LOCAL
0400   01  REC-BUFFER(A20/1:26)         /* max, record buffer passed to STPRBE
0410   01  REDEFINE REC-BUFFER          /* redefine this to get the definition
0420     02  INPUT-NAME   (A60)
0430     02  OUTPUT-NAME  (A60)
0440     02  KEYWORDS(A20/1:20)
0450   01  FUNC            (A4)
0460   01  SUB             (I2)
0470   01  SUB1            (I2)
0480   01  SUB2            (I2)
0490   01  W-UPPER        (A61)
0500   01  REDEFINE W-UPPER
0510     02  #UPPER       (A60)
```

```
0520     02  REDEFINE #UPPER
0530       03  CHAR   (A1/1:60)
0540   01  #KEYS     (A40/1:20)
0550 END-DEFINE
0560 *
0570 *  First, all procedures for this file must go through the audit procedure
0580 *  because our example requires a trace of all commands to this file.
0590 *
0600 CALLNAT 'SAMP0003' REQ-AREA RESP
0610 *
0620 * Since the record buffer information is available to us, we can now call
0630 * the record buffer extraction routine (STPRBE) to obtain the contents of
0640 * the buffer.
0650 *
0660 * Function 'GR' -- GET RB value using RB offset + length
0670 *     This enables the caller to obtain information based on a
0680 *     certain location; hence, RBE-OFFSET specifies the start
0690 *     position and RBE-LENGTH specifies the length.
0700 *
0710 MOVE 1   TO RBE-OFFSET             /* start at the beginning
0720 MOVE 520 TO RBE-LENGTH             /* for a max. length of 520 bytes
0730 MOVE 'GR' TO FUNC
0740 CALL 'STPRBE' 'GR' RBE-AREA REC-BUFFER(1)
0750 IF RBE-RESP NE 0
0760   PRINT *PROGRAM 'received an error from the STPRBE routine. Error:'
0770        RBE-ERROR 'subcode' RBE-SUBCODE 'for func GR'
0780   MOVE RBE-RESP TO RESP
0790   ESCAPE ROUTINE
0800 END-IF
0810 * PERFORM PRINT-REC-BUFFER         /* option to print the parameters
0820 *
0830 * Change all lowercase to UPPERcase
0840 *
0850 MOVE INPUT-NAME TO #UPPER
0860 *
0870 EXAMINE #UPPER AND TRANSLATE INTO UPPER CASE
0880 *
0890 MOVE #UPPER TO OUTPUT-NAME         /* save the uppercase name
0900 *
0910 FOR SUB 1 60                       /* loop to remove all special chars.
0920   IF CHAR(SUB) = MASK(S)
0930     MOVE ' ' TO CHAR(SUB)
0940     ESCAPE TOP
0950   END-IF
0960 END-FOR
0970 *
0980 * We are now ready to extract keywords from our name. This sample is
0990 * very basic and may be made as complex as required.
1000 * This routine assumes a max. length of 20 and a max. num. of 20 keywords
1010 *
1020 EXAMINE FULL W-UPPER FOR FULL ' A '   REPLACE ' '
1030 EXAMINE FULL W-UPPER FOR FULL ' AND ' REPLACE ' '
1040 EXAMINE FULL W-UPPER FOR FULL ' AS '  REPLACE ' '
1050 EXAMINE FULL W-UPPER FOR FULL ' AT '  REPLACE ' '
1060 EXAMINE FULL W-UPPER FOR FULL ' ARE ' REPLACE ' '
1070 EXAMINE FULL W-UPPER FOR FULL ' BE '  REPLACE ' '
1080 EXAMINE FULL W-UPPER FOR FULL ' DO '  REPLACE ' '
1090 EXAMINE FULL W-UPPER FOR FULL ' FOR ' REPLACE ' '
1100 EXAMINE FULL W-UPPER FOR FULL ' HERE ' REPLACE ' '
1110 EXAMINE FULL W-UPPER FOR FULL ' IF '  REPLACE ' '
1120 EXAMINE FULL W-UPPER FOR FULL ' IN '  REPLACE ' '
1130 EXAMINE FULL W-UPPER FOR FULL ' IS '  REPLACE ' '
```

```
1140 EXAMINE FULL W-UPPER FOR FULL ' IT '  REPLACE ' '
1150 EXAMINE FULL W-UPPER FOR FULL ' OF '  REPLACE ' '
1160 EXAMINE FULL W-UPPER FOR FULL ' ON '  REPLACE ' '
1170 EXAMINE FULL W-UPPER FOR FULL ' OR '  REPLACE ' '
1180 EXAMINE FULL W-UPPER FOR FULL ' TO '  REPLACE ' '
1190 EXAMINE FULL W-UPPER FOR FULL ' THE ' REPLACE ' '
1200 EXAMINE FULL W-UPPER FOR FULL ' TOO ' REPLACE ' '
1210 EXAMINE FULL W-UPPER FOR FULL ' WAS ' REPLACE ' '
1220 EXAMINE FULL W-UPPER FOR FULL ' WITH ' REPLACE ' '
1230 EXAMINE #UPPER FOR FULL ' ' REPLACE ',' /* put delimiters in the string
1240 *
1250 RESET KEYWORDS(*)
1260 STACK TOP DATA #UPPER                  /* now we will separate each word
1270 INPUT (AD=I IP=ON) #KEYS(01:03) / #KEYS(04:06) / #KEYS(07:09)
1280                 / #KEYS(10:12) /* #KEYS(13:15) / #KEYS(16:18)
1290                 / #KEYS(19:20)
1300 *
1310 MOVE 1 TO SUB2
1320 MOVE #KEYS(1) TO KEYWORDS(1)
1330 FOR SUB 2 20                           /* now we remove all duplicates
1340   FOR SUB1 1 SUB
1350     IF #KEYS(SUB) = KEYWORDS(SUB1)
1360       RESET #KEYS(SUB)
1370     END-IF
1380   END-FOR
1390   IF #KEYS(SUB) NE ' '
1400     ADD 1 TO SUB2
1410     MOVE #KEYS(SUB) TO KEYWORDS(SUB2)  /* and finally save the value
1420   END-IF
1430 END-FOR
1440 *
1450 * Function 'UR' -- Update RB value using RB offset + length
1460 *     This enables the caller to change information based on a
1470 *     certain location; hence, RBE-OFFSET specifies the start
1480 *     position and RBE-LENGTH specified the length.
1490 *
1500 * PERFORM PRINT-REC-BUFFER            /* print the final results
1510 MOVE 1   TO RBE-OFFSET                /* start at the beginning
1520 MOVE 520 TO RBE-LENGTH                /* for a max. length of 520 bytes
1530 MOVE 'UR' TO FUNC                     /* request to update all changes
1540 CALL 'STPRBE' 'UR' RBE-AREA REC-BUFFER(1)
1550 IF RBE-RESP NE 0
1560   PRINT *PROGRAM 'received an error from the STPRBE routine. Error:'
1570        RBE-ERROR 'subcode' RBE-SUBCODE 'for func UR'
1580   MOVE RBE-RESP TO RESP
1590   ESCAPE ROUTINE
1600 END-IF
1610 *
1620 * Return to the caller: everything went okay
1630 *
1640 ESCAPE ROUTINE
1650 *
1660 DEFINE SUBROUTINE PRINT-REC-BUFFER
1670 *----------------------------------------------------------------------*
1680 *
1690 *  For testing purposes, display the information returned from STPRBE
1700 *  This routine assumes a maximum of three subsystems running.
1710 *
1720 *----------------------------------------------------------------------*
1730   DECIDE ON FIRST VALUE OF RQ-TASK
1740     VALUE '01'
1750       WRITE (1) NOTITLE NOHDR (AD=L CD=TU)
```

```
1760           '>>>> RECORD BUFFER EXTRACTION: Function' FUNC '<<<<'
1770         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
1780         / '             ....' (TU) RBE-ADA-FIELD RBE-FIELD-OCC
1790         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR '<<<<<'
1800         / '  Message .......' (TU) RBE-MSG(AL=60)
1810         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
1820         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * * '
1830     VALUE '02'
1840       WRITE (2) NOTITLE NOHDR (AD=L CD=TU)
1850           '>>>> RECORD BUFFER EXTRACTION: Function' FUNC '<<<<'
1860         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
1870         / '             ....' (TU) RBE-ADA-FIELD  RBE-FIELD-OCC
1880         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR '<<<<<'
1890         / '  Message .......' (TU) RBE-MSG(AL=60)
1900         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
1910         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * * '
1920     VALUE '03'
1930       WRITE (3) NOTITLE NOHDR (AD=L CD=TU)
1940           '>>>> RECORD BUFFER EXTRACTION: Function' FUNC '<<<<'
1950         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
1960         / '             ....' (TU) RBE-ADA-FIELD  RBE-FIELD-OCC
1970         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR '<<<<<'
1980         / '  Message .......' (TU) RBE-MSG(AL=60)
1990         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
2000         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * * '
2010     NONE
2020       WRITE NOTITLE NOHDR (AD=L CD=TU)
2030           '>>>> RECORD BUFFER EXTRACTION: function' FUNC '<<<<'
2040         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
2050         / '             ....' (TU) RBE-ADA-FIELD  RBE-FIELD-OCC
2060         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR'<<<<<'
2070         / '  Message .......' (TU) RBE-MSG(AL=60)
2080         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
2090         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * * '
2100   END-DECIDE
2110 *
2120 END-SUBROUTINE
2130 *
2140 END
```

SAMP0002

```
0010 ************************************************************************
0020 *  Application: Adabas Triggers
0030 *   Subprogram: SAMP0002
0040 *     Function: Sample routine of processing by a stored procedure
0050 *               The requirement is to audit all commands for a file
0060 *               by writing out an audit record to a file/printer.
0070 *               For this example, the audit is the Natural system file.
0080 * Trigger Defn: Trigger Information
0090 *                 File Number ....... 11
0100 *                 File Name ......... CONTACTS
0110 *                 Command Type ...... ** All Command **
0120 *                 Long Field Name ... ** Any Field **
0130 *                 Adabas Field ...... **
0140 *                 Field Prty/Seq .... ___
0150 *               Procedure Information
0160 *                 Name (Subpgm)...... SAMP0002
0170 *                 Pre Cmd Select .... Y  (Pre)
0180 *                 Trigger Type ...... A  (Asynchronous)
0190 *                 CALLNAT Params .... C  (Cntl Info + Resp)
```

```
0200 *                 RecBuffer Access .. N  (No RecBuff Access)
0210 *
0220 *       AUTHOR: Adabas Development
0230 *         DATE: December 1995
0240 ************************************************************************
0250 DEFINE DATA PARAMETER USING STPAPARM
0260            LOCAL     USING STPLCB /* DSECT of the Adabas control block
0270            LOCAL
0280  01  #SRCID     (A18)             /* key of the audit record
0290  01  REDEFINE #SRCID
0300    02  SRC-LIB   (A8)             /* to be placed on a Natural system file
0310    02  SRC-PGM   (A8)
0320    02  SRC-SEQ   (B2)
0330  01  #DATE       (A8)
0340  01  #TIME       (A8)
0350  01  LOG-AREA VIEW OF SYSTEM2     /* write information to the FNAT file
0360    02  SRCID
0370    02  SRCTX      (1)
0380  01  W-USERID   (A28)             /* user ID from originating command
0390  01  REDEFINE W-USERID
0400    02  W-F1     (A20)
0410    02  W-USER    (A8)             /* TP USID of the user ID
0420  01  #TEXT (A72)                  /* text message to be written
0430  01  REDEFINE #TEXT
0440    02  TX-LNO    (B2)
0450    02  TX-F1     (A1)
0460    02  TX-DATE   (A8)
0470    02  TX-F2     (A1)
0480    02  TX-TIME   (A5)
0490    02  TX-F3     (A1)
0500    02  TX-USER   (A8)
0510    02  TX-F4     (A1)
0520    02  TX-CMD    (A2)
0530    02  TX-F5     (A1)
0540    02  TX-PRE    (A3)
0550    02  TX-F6     (A1)
0560    02  TX-FNR    (N3)
0570    02  TX-F7     (A1)
0580    02  TX-RBL    (N4)
0590    02  TX-F8     (A1)
0600    02  TX-SYNC   (A5)
0610    02  TX-F9     (A1)
0620    02  TX-TASK   (A2)
0630    02  TX-F10    (A1)
0640    02  TX-FIELD  (A2)
0650    02  TX-F11    (A1)
0660    02  TX-PROC   (A8)
0670    02  TX-F12    (A1)
0680    02  TX-USR2   (A8)
0690 END-DEFINE
0700 *
0710 ASSIGN #SRCID =  'AUDIT   LOGINFO' /* set the target lib and pgm name
0720 MOVE H'0000' TO SRC-SEQ
0730 *
0740 MOVE RQ-CB   TO CB                 /* move ACB into our CB layout
0750 MOVE H'0010' TO TX-LNO             /* line number of Natural source
0760 MOVE *DATE   TO TX-DATE
0770 MOVE *TIMX   TO TX-TIME
0780 MOVE RQ-USERID TO W-USERID         /* user ID of the command
0790 MOVE W-USER  TO TX-USER            /* may be a batch user
0800 IF NOT TX-USER = MASK(PPPPPPPP)    /* printable user ID?
0810  MOVE RQ-USER TO TX-USER           /* no, so use the jobname or TPname
```

```
0820 END-IF
0830 MOVE RQ-CMD  TO TX-CMD            /* information from the CB layout
0840 MOVE RQ-FNR  TO TX-FNR
0850 MOVE RQ-TASK TO TX-TASK           /* subsystem number
0860 IF RQ-LENGTH > 9999               /* exceed max. size in audit message?
0870  MOVE 9999 TO TX-RBL
0880 ELSE
0890  MOVE RQ-LENGTH TO TX-RBL         /* the real record buffer length
0900 END-IF
0910 MOVE *PROGRAM  TO TX-PROC         /* originating procedure. This subpgm
0920 IF RQ-TTYPE = 'P'                 /* trigger type
0930  MOVE 'Pre' TO TX-PRE
0940 ELSE
0950  MOVE 'Pos' TO TX-PRE
0960 END-IF
0970 IF RQ-SYNC = 'A'                  /* processing type
0980  MOVE 'ASync' TO TX-SYNC
0990 ELSE
1000  IF RQ-PARTIC = 'P'               /* trigger logic type
1010    MOVE 'Part'  TO TX-SYNC
1020  ELSE
1030    MOVE 'Non-P' TO TX-SYNC
1040  END-IF
1050 END-IF
1060 *
1070 * Now we do some logic to write the information out to a report
1080 * Here we support up to five subsystems
1090 * Contents are a one-line display to minimize output
1100 *
1110 DECIDE ON FIRST VALUE OF RQ-TASK
1120    VALUE '01'
1130      DISPLAY (1) NOTITLE (AD=L CD=TU)
1140            'Procedure' *PROGRAM
1150            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1160            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1170            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1180            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1190    VALUE '02'
1200      DISPLAY (2) NOTITLE (AD=L CD=TU)
1210            'Procedure' *PROGRAM
1220            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1230            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1240            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1250            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1260    VALUE '03'
1270      DISPLAY (3) NOTITLE (AD=L CD=TU)
1280            'Procedure' *PROGRAM
1290            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1300            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1310            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1320            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1330    VALUE '04'
1340      DISPLAY (4) NOTITLE (AD=L CD=TU)
1350            'Procedure' *PROGRAM
1360            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1370            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1380            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1390            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1400    NONE
1410      DISPLAY (5) NOTITLE (AD=L CD=TU)
1420            'Procedure' *PROGRAM
1430            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
```

```
1440            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1450            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1460            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1470  END-DECIDE
1480 *
1490 * Finally, we write this information to a 'audit' File. In this case, we
1500 * use the Natural FNAT file for simplicity. Realistically, a separate
1510 * 'audit' file should be used.
1520 *
1530  MOVE #TEXT TO LOG-AREA.SRCTX (1.1)
1540  MOVE H'0001' TO SRC-SEQ
1550  ASSIGN LOG-AREA.SRCID = #SRCID
1560  STORE LOG-AREA
1570  END TRANSACTION                    /* required for non-participating
1580 *                                   /* and asynchronous triggers
1590 *
1600 END
```

SAMP0003

```
0010 ************************************************************************
0020 *  Application: Adabas Triggers
0030 *   Subprogram: SAMP0003
0040 *     Function: Sample routine of processing by a stored procedure
0050 *               The requirement is to audit  all commands for a file
0060 *               by writing out an audit record to a file/printer.
0070 *               For this example, the audit is the Natural system file.
0080 *
0090 *               This routine is called by participating triggers and
0100 *               will contain no ET logic; hence, it must have been
0110 *               invoked as a result of a update/delete/store command
0120 * Trigger Defn: None because it will be invoked directly from another
0130 *               procedure. In this case SAMP0001.
0140 *
0150 *       Author: Adabas Development
0160 *         Date: December 1995
0170 ************************************************************************
0180 DEFINE DATA PARAMETER USING STPAPARM
0190            LOCAL     USING STPLCB /* DSECT of the Adabas control block
0200            LOCAL
0210  01  #SRCID     (A18)             /* key of the audit record
0220  01  REDEFINE #SRCID
0230    02  SRC-LIB   (A8)             /* to be placed on a Natural system file
0240    02  SRC-PGM   (A8)
0250    02  SRC-SEQ   (B2)
0260  01  #DATE       (A8)
0270  01  #TIME       (A8)
0280  01  LOG-AREA VIEW OF SYSTEM2     /* write information to the FNAT file
0290    02  SRCID
0300    02  SRCTX      (1)
0310  01  W-USERID   (A28)             /* user ID from originating command
0320  01  REDEFINE W-USERID
0330    02  W-F1     (A20)
0340    02  W-USER    (A8)             /* TP USID of the user ID
0350  01  #TEXT (A72)                  /* text message to be written
0360  01  REDEFINE #TEXT
0370    02  TX-LNO    (B2)
0380    02  TX-F1     (A1)
0390    02  TX-DATE   (A8)
0400    02  TX-F2     (A1)
0410    02  TX-TIME   (A5)
```

```
0420    02  TX-F3     (A1)
0430    02  TX-USER   (A8)
0440    02  TX-F4     (A1)
0450    02  TX-CMD    (A2)
0460    02  TX-F5     (A1)
0470    02  TX-PRE    (A3)
0480    02  TX-F6     (A1)
0490    02  TX-FNR    (N3)
0500    02  TX-F7     (A1)
0510    02  TX-RBL    (N4)
0520    02  TX-F8     (A1)
0530    02  TX-SYNC   (A5)
0540    02  TX-F9     (A1)
0550    02  TX-TASK   (A2)
0560    02  TX-F10    (A1)
0570    02  TX-FIELD  (A2)
0580    02  TX-F11    (A1)
0590    02  TX-PROC   (A8)
0600    02  TX-F12    (A1)
0610    02  TX-USR2   (A8)
0620 END-DEFINE
0630 *
0640 ASSIGN #SRCID =  'AUDIT   LOGINFO' /* set the target lib and program name
0650 MOVE H'0000' TO SRC-SEQ
0660 *
0670 MOVE RQ-CB   TO CB                 /* move ACB into the CB layout
0680 MOVE H'0010' TO TX-LNO             /* line number of Natural source
0690 MOVE *DATE   TO TX-DATE
0700 MOVE *TIMX   TO TX-TIME
0710 MOVE RQ-USERID TO W-USERID         /* user ID of the command
0720 MOVE W-USER  TO TX-USER            /* may be a batch user
0730 MOVE RQ-USER TO TX-USR2            /* jobname or TPname
0740 MOVE RQ-CMD  TO TX-CMD             /* information from the CB layout
0750 MOVE RQ-FNR  TO TX-FNR
0760 MOVE RQ-TASK TO TX-TASK            /* subsystem number
0770 IF RQ-LENGTH > 9999                /* exceed max. size in audit message?
0780   MOVE 9999 TO TX-RBL
0790 ELSE
0800   MOVE RQ-LENGTH TO TX-RBL         /* the real record buffer length
0810 END-IF
0820 MOVE *PROGRAM  TO TX-PROC          /* originating procedure. This subpgm
0830 IF RQ-TTYPE = 'P'                  /* trigger type
0840   MOVE 'Pre' TO TX-PRE
0850 ELSE
0860   MOVE 'Pos' TO TX-PRE
0870 END-IF
0880 IF RQ-SYNC = 'A'                   /* processing type
0890   MOVE 'ASync' TO TX-SYNC
0900 ELSE
0910   IF RQ-PARTIC = 'P'               /* trigger logic type
0920     MOVE 'Part'  TO TX-SYNC
0930   ELSE
0940     MOVE 'Non-P' TO TX-SYNC
0950   END-IF
0960 END-IF
0970 *
0980 * Now we do some logic to write the information out to a report
0990 * Here we support up to five subsystems
1000 * Contents are a one-line display to minimize output
1010 *
1020 DECIDE ON FIRST VALUE OF RQ-TASK
1030    VALUE '01'
```

```
1040      DISPLAY (1) NOTITLE (AD=L CD=TU)
1050            'Procedure' *PROGRAM
1060            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1070            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1080            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1090            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1100            'Usr2' (TU) TX-USR2
1110    VALUE '02'
1120      DISPLAY (2) NOTITLE (AD=L CD=TU)
1130            'Procedure' *PROGRAM
1140            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1150            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1160            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1170            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1180            'Usr2' (TU) TX-USR2
1190    VALUE '03'
1200      DISPLAY (3) NOTITLE (AD=L CD=TU)
1210            'Procedure' *PROGRAM
1220            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1230            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1240            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1250            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1260            'Usr2' (TU) TX-USR2
1270    VALUE '04'
1280      DISPLAY (4) NOTITLE (AD=L CD=TU)
1290            'Procedure' *PROGRAM
1300            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1310            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1320            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1330            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1340            'Usr2' (TU) TX-USR2
1350    NONE
1360      DISPLAY (5) NOTITLE (AD=L CD=TU)
1370            'Procedure' *PROGRAM
1380            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1390            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1400            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1410            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1420            'Usr2' (TU) TX-USR2
1430  END-DECIDE
1440 *
1450 * Finally we write this info to a 'audit' file. In this case, we use the
1460 * Natural FNAT file for simplicity. Realistically, a separate 'audit' file
1470 * should be used. End Transaction (ET) must  not be issued because this
1480 * will be controlled by the application and not the trigger procedure.
1490 *
1500  MOVE #TEXT TO LOG-AREA.SRCTX (1.1)
1510  MOVE H'0001' TO SRC-SEQ
1520  ASSIGN LOG-AREA.SRCID = #SRCID
1530  STORE LOG-AREA
1540 *
1550 END
```

SAMP0004

```
0010 ************************************************************************
0020 *  Application: Adabas Triggers
0030 *   Subprogram: SAMP0004
0040 *     Function: Sample routine of processing by a stored procedure
0050 *               referential integrity - RESTRICT
0060 *               (assume that the primary key is on the EMPLOYEES file and
```

```
0070 *                the foreign key on the VEHICLES + MISCELLANEOUS files).
0080 * Trigger Defn: Definition on the trigger file is as follows:
0090 *                 File Number ....... 3
0100 *                 File Name ......... VEHICLES-FILE
0110 *                 Command Type ...... Delete
0120 *                 Long Field Name ... ** Any Field **
0130 *                 Adabas Field ...... **
0140 *                 Field Prty/Seq .... ___
0150 *               Procedure Information
0160 *                 Name (Subpgm)...... SAMP0004
0170 *                 Pre Cmd Select .... Y  (Pre)
0180 *                 Trigger Type ...... N  (Non-Participating)
0190 *                 CALLNAT Params .... C  (Cntl Info + Resp)
0200 *                 RecBuffer Access .. N  (No RecBuff Access)
0210 *
0220 *      Invoked: Invoked with deletes from VEHICLES/MISCELLANEOUS files
0230 *               Sample Routine: SAMPREF1
0240 *       Author: Adabas Development
0250 *         Date: December 1995
0260 ************************************************************************
0270 DEFINE DATA PARAMETER USING STPAPARM
0280           LOCAL
0290   01  VEHICLES  VIEW OF VEHICLES
0300     02  PERSONNEL-ID               /* foreign key: field AC
0310   01  MISCELLANEOUS VIEW OF MISCELLANEOUS
0320     02  PERSONNEL-ID               /* foreign key: field CA
0330   01  EMPLOYEES VIEW OF EMPLOYEES
0340     02  PERSONNEL-ID               /* primary key: field AA
0350   01  #FILE        (P5)
0360   01  #ISN        (P10)
0370   01  #PERS-NUM    (A8)
0380   01  CONTRL-BLK  (A80)
0390   01  REDEFINE CONTRL-BLK
0400     02  CB-FIL1   (A12)
0410     02  CB-ISN     (B4)
0420 END-DEFINE
0430 *
0440 *  First we extract the foreign key information
0450 *  i.e., get the ISN of the record in the ACB and read this record
0460 *    to extract the required information; i.e., the foreign key info.
0470 *    NOTE: With a delete, no data is passed in the record buffer.
0480 *
0490 MOVE RQ-CB  TO CONTRL-BLK         /* get the ACB of the originating cmd
0500 MOVE RQ-FNR TO #FILE              /* find out which file has the delete
0510 MOVE CB-ISN TO #ISN               /* ISN of the record to be deleted
0520 *
0530 IF #FILE = 3                      /* identify the file: Vehicles
0540   DO
0550     GET VEHICLES #ISN             /* get the value of the foreign key
0560     MOVE PERSONNEL-ID(0550) TO #PERS-NUM    /* get the key
0570   DOEND
0580 ELSE
0590   IF #FILE = 2                    /* or the Miscellaneous file
0600     DO
0610       GET MISCELLANEOUS #ISN      /* get the value of the foreign key
0620       MOVE PERSONNEL-ID(0610) TO #PERS-NUM  /* get the key
0630     DOEND
0640   ELSE                            /* a check for the unexpected...
0650     DO                            /* a trigger may have been defined wrong
0660       MOVE 913 TO RQ-RESP         /* either ignore or return an error
0670       ESCAPE ROUTINE              /* and exit
0680     DOEND
```

```
0690 *
0700 RESET RQ-RESP
0710 *
0720 * Now we check the primary file to see if the value exists. If yes
0730 * then we cannot allow this deletion; hence, we prevent any deletions
0740 * of the foreign key files if a record with the same key exists on the
0750 * primary file.
0760 *
0770 * NOTE: With the setting of RESP, consideration should be given to
0780 *       ambiguities. While the command will receive a response 155
0790 *       (pre-trigger) or 156 (post-trigger), the additions field will
0800 *       contain the error returned from this procedure. The value
0810 *       could be in the form of an Adabas response (1-255) or a
0820 *       Natural error (e.g., 954 or 935 or 3009); therefore, a
0830 *       user-specified error from the procedure should be something
0840 *       outside these ranges........for simplicity.
0850 *
0860 FIND EMPLOYEES WITH PERSONNEL-ID = #PERS-NUM
0870   MOVE 901 TO RESP                /* it does: delete may not be done
0880   ESCAPE ROUTINE
0890 CLOSE LOOP(0860)
0900 *
0910 END
```

SAMP0005

```
0010 ************************************************************************
0020 *  Application: ADASTP
0030 *   Subprogram: SAMP0005
0040 *     Function: Sample routine of processing by a stored procedure
0050 *               referential integrity - CASCADE
0060 *               (assume that the primary key is on the EMPLOYEES file
0070 *               and foreign keys on the VEHICLES + MISCELLANEOUS files).
0080 * Trigger Defn: Definition on the trigger file is as follows:
0090 *                 File Number ....... 4
0100 *                 File Name ......... EMPLOYEES
0110 *                 Command Type ...... Update
0120 *                 Long Field Name ... PERSONNEL-ID
0130 *                 Adabas Field ...... AA
0140 *                 Field Prty/Seq .... 010
0150 *               Procedure Information
0160 *                 Name (Subpgm)...... SAMP0005
0170 *                 Pre Cmd Select .... Y  (Pre)
0180 *                 Trigger Type ...... P  (Participating)
0190 *                 CALLNAT Params .... C  (Cntl Info + Resp)
0200 *                 RecBuffer Access .. A  (May be Accessed)
0210 *
0220 *      Invoked: Invoked with updates to the EMPLOYEES PERSONNEL-ID
0230 *               Sample routine: SAMPREF2
0240 *       Author: Adabas Development
0250 *         Date: December 1995
0260 ************************************************************************
0270 DEFINE DATA PARAMETER USING STPAPARM
0280            LOCAL     USING STPLRBE      /* parameters for call to STPRBE
0290            LOCAL
0300  01  EMPLOYEES VIEW OF EMPLOYEES
0310    02  PERSONNEL-ID                    /* primary key: field AC
0320  01  MISCELLANEOUS VIEW OF MISCELLANEOUS
0330    02  PERSONNEL-ID                    /* foreign key: field CA
0340  01  VEHICLES  VIEW OF VEHICLES
0350    02  PERSONNEL-ID                    /* foreign key: field AA
```

```
0360  01  FUNC         (A4)
0370  01  #ISN        (P10)
0380  01  #PERS-NUM    (A8)
0390  01  CONTRL-BLK  (A80)
0400  01  REDEFINE CONTRL-BLK
0410    02  CB-FIL1   (A12)
0420    02  CB-ISN     (B4)
0430 END-DEFINE
0440 *
0450 *  First we extract the foreign key information
0460 *  There are two ways to pick this up:
0470 *
0480 *      1) Since the value is in the record buffer, we can use STPRBE to
0490 *         extract the required information; i.e., the primary key
0500 *         information. There are three ways to do this...in this case:
0510 *
0520 *             A) identify the field by its long name; i.e., PERSONNEL-ID
0530 *             B) identify the field by its short name; i.e., AA
0540 *             C) identify the location and length in the record buffer
0550 *
0560 *      2) Get the ISN of the record in the ACB and read this record to
0570 *         extract the required information; i.e., the primary key
0580 *         information. However, this is the old value and cannot be used
0590 *         in this example.
0600 *
0610 *
0620 * OPTION 1A
0630 *
0640 * Function 'GV' -- GET field value using the long field name
0650 *     This enables the caller to obtain information about a specific
0660 *     field which is determined according to the long field name
0670 *     passed in the parameters to STPRBE.
0680 *
0690 RESET #PERS-NUM
0700 MOVE 'GV'           TO FUNC
0710 MOVE 'PERSONNEL-ID' TO RBE-FIELD-NAME /* and identify the corresponding
0720 *                                        field for this file
0730 MOVE 8              TO RBE-LENGTH     /* default or give override length
0740 *                                     /* length in FB could have been used
0750 CALL 'STPRBE' FUNC RBE-AREA #PERS-NUM
0760 PRINT *PROGRAM 'Option 1A returned ..' #PERS-NUM 'resp' RBE-RESP
0770 IF RBE-RESP NE 0                      /* successful ?
0780  DO
0790    PRINT *PROGRAM 'received an error from the STPRBE routine. Error:'
0800          RBE-ERROR 'subcode' RBE-SUBCODE 'for func GV'
0810    MOVE RBE-RESP TO RESP              /* indicate this
0820    ESCAPE ROUTINE                     /* and exit
0830  DOEND
0840 *
0850 * OPTION 1B
0860 *
0870 * Function 'GV' -- GET field value using short field name
0880 *     This enables the caller to obtain information about a specific
0890 *     field which is determined according to the short field name
0900 *     passed in the parameters to STPRBE.
0910 *     NOTE: '**' in field name means user-supplied details in short name
0920 *
0930 RESET #PERS-NUM
0940 MOVE 'GV'     TO FUNC
0950 MOVE '**'     TO RBE-FIELD-NAME    /* special notation for this request
0960 MOVE RQ-FIELD TO RBE-ADA-FIELD     /* get field name that fired the
0970 *                                     trigger from the parm area...OR.....
```

```
0980 IF NOT (RQ-FIELD = 'AA')           /* if we know the field....
0990   MOVE 'AA'   TO RBE-ADA-FIELD     /* identify the specific field name
1000 MOVE 8        TO RBE-LENGTH        /* for a maximum length of 8 bytes
1010 CALL 'STPRBE' FUNC RBE-AREA #PERS-NUM
1020 PRINT *PROGRAM 'Option 1B returned ..' #PERS-NUM 'resp' RBE-RESP
1030 IF RBE-RESP NE 0                   /* successful
1040  DO
1050    PRINT *PROGRAM 'received an error from the STPRBE routine. Error:'
1060          RBE-ERROR 'subcode' RBE-SUBCODE 'for func GV'
1070    MOVE RBE-RESP TO RESP           /* indicate this
1080    ESCAPE ROUTINE                  /* and exit
1090  DOEND
1100 *
1110 * OPTION 1C
1120 *
1130 * Function 'GR' -- GET RB value using RB offset + length
1140 *     This enables the caller to obtain information based on a
1150 *     certain location; hence, RBE-OFFSET specifies the start
1160 *     position and RBE-LENGTH specifies the length.
1170 *
1180 RESET #PERS-NUM
1190 MOVE 'GR' TO FUNC
1200 MOVE 1    TO RBE-OFFSET            /* start at the beginning
1210 MOVE 8    TO RBE-LENGTH            /* for a max. length of 50 bytes
1220 CALL 'STPRBE' FUNC RBE-AREA #PERS-NUM
1230 PRINT *PROGRAM 'Option 1C returned ..' #PERS-NUM 'resp' RBE-RESP
1240 IF RBE-RESP NE 0
1250  DO
1260    PRINT *PROGRAM 'received an error from the STPRBE routine. Error:'
1270          RBE-ERROR 'subcode' RBE-SUBCODE 'for func GR'
1280    MOVE RBE-RESP TO RESP
1290    ESCAPE ROUTINE
1300  DOEND
1310 *
1320 *  NOTE: Only one of the options need be used to extract the value
1330 *
1340 RESET RQ-RESP
1350 *
1360 * Now, we read the original record, which is not yet changed; hence the
1370 * reason for setting this up as a pre-trigger, to see if the value
1380 * (PERSONNEL-ID in this case) has changed.
1390 *
1400 MOVE RQ-CB  TO CONTRL-BLK          /* get the original ACB of the A1/4
1410 MOVE CB-ISN TO #ISN                /* extract the ISN of the record
1420 GET EMPLOYEES #ISN                 /* read the, so far, unchanged data
1430 IF PERSONNEL-ID(1420) = #PERS-NUM  /* have the numbers changed?
1440   ESCAPE ROUTINE                   /* no, then exit
1450 *
1460 * Now that we have observed that the primary key has changed, we must
1470 * read all the files with a foreign key and CASCADE the update.
1480 *
1490 FIND VEHICLES WITH PERSONNEL-ID = PERSONNEL-ID(1420) /* Vehicles file
1500  ASSIGN PERSONNEL-ID(1490) = #PERS-NUM
1510  UPDATE (1490)
1520 CLOSE LOOP(1490)
1530 *
1540 FIND MISCELLANEOUS WITH PERSONNEL-ID = PERSONNEL-ID(1420) /* Misc file
1550  ASSIGN PERSONNEL-ID(1540) = #PERS-NUM
1560  UPDATE (1540)
1570 CLOSE LOOP(1540)
1580 *
1590 * Issuing an ET now, is not valid with a participating trigger because
```

```
1600 * the originating command (A1/Update) has not yet been executed and
1610 * the originating user expects to do the ET once the update is complete.
1620 * If this ET were done here, the A1/4 (pre-trigger) would receive a
1630 * response 144 because the ISN would be released. If the originating
1640 * user had to do other updates, then a misplaced ET (End Transaction)
1650 * could cause a loss of data integrity across the files.
1660 *
1670 END
```

SAMPREF1

```
0010 ************************************************************************
0020 * Application: Adabas Triggers
0030 *     Program: SAMPREF1 - Example of referential integrity (restrict)
0040 *    Function: SPT routine to delete records from the Vehicles file
0050 *              Invoked with a delete trigger as shown below:
0060 *
0070 *              Trigger Information
0080 *                File Number ....... 3
0090 *                File Name ......... VEHICLES-FILE
0100 *                Command Type ...... Delete
0110 *                Long Field Name ... ** Any Field **
0120 *                Adabas Field ...... **
0130 *                Field Prty/Seq .... ___
0140 *              Procedure Information
0150 *                Name (Subpgm)...... SAMP0004
0160 *                Pre Cmd Select .... Y  (Pre)
0170 *                Trigger Type ...... N  (Non-Participating)
0180 *                CALLNAT Params .... C  (Cntl Info + Resp)
0190 *                RecBuffer Access .. N  (No RecBuff Access)
0200 *
0210 ************************************************************************
0220 DEFINE DATA LOCAL
0230   01  #NUMBER    (A8)
0240   01  VEHICLES VIEW OF VEHICLES
0250     02  PERSONNEL-ID
0260 END-DEFINE
0270 *
0280 INPUT (AD=TMIL'_' CD=NE)
0290      'Trigger Example for Referential Integrity - RESTRICT' (YEI)
0300   // 'Enter Personnel Number ..' (TU) #NUMBER
0310 *
0320 IF #NUMBER = MASK('.')                    /* exit?
0330   STOP                                    /* yes
0340 IF #NUMBER = ' '                          /* a number must be specified
0350   REINPUT 'Invalid Number specified'
0360 *
0370 FIND VEHICLES WITH PERSONNEL-ID = #NUMBER /* find the record to be deleted
0380   DELETE(0370)                            /* issue the Delete request
0390   END TRANSACTION                         /* finalize the delete
0400   REINPUT 'Record has now been deleted'   /* confirm and restart
0410 CLOSE LOOP
0420 IF *NUMBER(0370) = 0                      /* validate existence of number
0430   REINPUT 'Invalid Personnel Number specified'
0440 *
0450 * Below, any error handling may be done. With a trigger, a procedure
0460 * could return a non-zero response. This would result in the trigger
0470 * command (the Delete in this case) receiving a response 155. Pre-triggers
0480 * receive a response 155 and post-triggers receive a response 156.
0490 *
0500 ON ERROR                          /* handle any errors from the trigger
```

```
0510   DO
0520     BACKOUT TRANSACTION            /* release the held record/ISN
0530     INPUT (AD=O CD=YE) 8X '*** Warning ***' (REI)
0540       // 'Personnel Number' (YE) #NUMBER 'NOT deleted' (YEI)
0550        / 'Response' (YE) *ERROR-NR 'received for this request' (YE)
0560       // 4X 'Press Enter to continue' (REI)
0570     STACK TOP COMMAND *PROGRAM     /* return to start of this routine
0580     STOP
0590   DOEND
0600 END
```

SAMPREF2

```
0010 ************************************************************************
0020 * Application: Adabas Triggers
0030 *     Program: SAMPREF2 - Example of referential integrity (Cascade)
0040 *    Function: SPT routine to update records on the Employees file
0050 *              Invoked with an update trigger as shown below:
0060 *
0070 *              Trigger Information
0080 *                File Number ....... 4
0090 *                File Name ......... EMPLOYEES
0100 *                Command Type ...... Update
0110 *                Long Field Name ... PERSONNEL-ID
0120 *                Adabas Field ...... AA
0130 *                Field Prty/Seq .... 10_
0140 *              Procedure Information
0150 *                Name (Subpgm)...... SAMP0005
0160 *                Pre Cmd Select .... Y  (Pre)
0170 *                Trigger Type ...... P  (Participating)
0180 *                CALLNAT Params .... C  (Cntl Info + Resp)
0190 *                RecBuffer Access .. A  (May be Accessed)
0200 *
0210 ************************************************************************
0220 DEFINE DATA LOCAL
0230   01  #NUMBER    (A8)
0240   01  EMPLOYEES VIEW OF EMPLOYEES
0250     02  PERSONNEL-ID
0260     02  FIRST-NAME
0270     02  NAME
0280     02  MIDDLE-NAME
0290 END-DEFINE
0300 *
0310 REPEAT
0320 *
0330 INPUT (AD=TMIL'_' CD=NE)
0340       'Trigger Example for Referential Integrity - CASCADE' (YEI)
0350    // 'Enter Personnel Number ..' (TU) #NUMBER
0360 *
0370 IF #NUMBER = MASK('.')                    /* exit ?
0380   STOP                                    /* yes
0390 *
0400 IF #NUMBER = ' '                          /* a number must be specified
0410   REINPUT 'Invalid Number specified'
0420 *
0430 FIND EMPLOYEES WITH PERSONNEL-ID = #NUMBER /* read the record
0440   INPUT (AD=MIL CD=NE)                    /* show data for doing updates
0450         'Enter Employee Details Below for Update:-' (YEI)
0460     // 'Personnel Number ...' (TU) PERSONNEL-ID
0470      / 'Last Name ..........' (TU) NAME
0480      / 'First Name .........' (TU) FIRST-NAME
```

```
0490      / 'Middle Name ........' (TU) MIDDLE-NAME
0500 *
0510 * Validation of the changes may now be done as required
0520 *
0530   UPDATE(0430)                     /* make the database changes
0540   END TRANSACTION                  /* and finalize them
0550   ESCAPE BOTTOM
0560 CLOSE LOOP
0570 IF *NUMBER(0430) = 0
0580   REINPUT 'Invalid Personnel Number specified'
0590 ELSE
0600   INPUT NO ERASE ////// 4X 'Record has now been updated' (YEI)
0610 *
0620 CLOSE LOOP(0310)                   /* repeat loop
0630 *
0640 * Below, any error handling may be done. With a trigger, a procedure
0650 * could return a non-zero response. This would result in the trigger
0660 * command (the update in this case) receiving a response 155. Pre-triggers
0670 * receive a response 155 and post-triggers receive a response 156.
0680 *
0690 ON ERROR                           /* handle any errors from the trigger
0700   DO
0710     BACKOUT TRANSACTION            /* release the held record/ISN
0720     INPUT (AD=O CD=YE) 8X '*** Warning ***' (REI)
0730       // 'Personnel Number' (YE) #NUMBER 'NOT Updated' (YEI)
0740        / 'Response' (YE) *ERROR-NR 'received for this request' (YE)
0750       // 4X 'Press Enter to continue' (REI)
0760     STACK TOP COMMAND *PROGRAM     /* return to start of this routine
0770     STOP
0780   DOEND
0790 END
```