TABLES/AS

Technical Guide

May 2005

This edition applies to Release 5.1 of **TABLES/AS**

TABLES/AS TECHNICAL GUIDE

TABLE OF CONTENTS

Preface

This manual is intended to serve as the technical guide to the TABLES/AS Management System.

This manual should be used in conjunction with the following related publications:

TABLES/DM DB2 Table Definition & Migration Reference Manual
TABLES/AS Generalized Online Maintenance Reference Manual
The TABLES Memory Manager Reference Manual
TABLES/DM DB2 Table Definition & Migration Installation Guide
TABLES/AS Generalized Online Maintenance Installation Guide
TABLES/MM The TABLES Memory Manager Installation Guide

All JCL examples use the following data set name qualifiers:

Qualifier	Data Set Category
SSI.TEST	**TABLES** package libraries
SSI.TEST	**TABLES** test files

Section 1

TABLES/AS Technical Guide

Security

Table Security

The **TABLES/AS** system runs as a standard application. All security considerations that apply to an in-house application, will also apply to **TABLES/AS**.

Screen Development Security

Screens generated by **TABLES/AS** may provide additional security in several ways.

The definitions (screen layout, field edits, relational edits, and mapping) of a screen may be protected from changes.

This is provided by means of the control table SCRLOK. A record in this table is automatically created when a new screen is added in **TABLES/AS**. The user may protect a screen using the **TABLES/AP** processing menu, after a screen is fully defined and tested. The protection of the screen is made by entering a lock indicator of Y for each of the four major functions of screen programs:

LOCKDESN	= SCREEN DESIGN
LOCKEDIT	= FIELD EDITING
LOCKREL	= RELATIONAL EDITING
LOCKCONV	= MAPPING

For authid qualification of the control table SCRLOK, please check with your software support.

Screen Processing Security

1. The Screen Processor can be driven directly by its own supplied transaction code (TS09) or user-assigned transaction codes which can be protected from certain terminals and users, using your standard security interfaces.

2. For DB2 tables no USERID check is performed, it is left for DB2 to verify. The Screen Processor may require a user ID for non DB2 table processing.

3. **TABLES/AS** makes a distinction between a development environment and production environment. **TABLES/AS** assumes a development environment when entering the Screen Processor from the **TABLES/AS** main menu. All security identified above is in effect when entering the Screen Processor any other way.

4. Additional security may be designed by users on any screen to be processed. This can be done by establishing password fields on the screen and defining field and relational level edits. For example, assume that access to a table needs to be allowed to selected people from selected departments in which only some of the people are authorized to perform updates. This requirement may be implemented follows:

   - Define a password field with a non-displayable attribute and select editing option with discrete values.

   - Define a department number field with a non-displayable attribute and select editing option with discrete values.

   - Define relational editing using password, department number and function code fields to require password input for updates.

Screen Editor and SQL Load Modules Security

Screen load modules, editor load modules and static SQL modules are dynamically accessed by the mainline screen processor program, TSxM0900.[1] The security at the mainline program will always apply to the sub-modules. The mainline programs and sub-programs may be protected through security systems such as RACF, ACF2, and Top Secret, etc.

[1] where x = I for IMS
x = C for CICS

Section 2

TABLES/AS Technical Guide

Table Access

Interfaces To Application Programs

DB2 Modules	VSAM Modules
DB2TABLP (Batch)	**VSMFILEP** (Batch)
DB2TABLO (Online and batch IMS)	**V2STABLP** (CICS)
V2STABLP (CICS)	

Table searches are performed in memory using a binary search on ORDER BY fields.

DB2TABLP, DB2TABLO, V2STABLP

DB2TABLP, DB2TABLO and V2STABLP provide an efficient and simple way of accessing DB2 tables. Use of this facility eliminates the need to load DB2 tables in working storage or to provide logic to search from them. Additionally, the overhead of a large number of SQL calls is also eliminated. Other benefits result from the elimination of DB2 pre-compilation of programs that access tables in a read-only mode.

Any application using these interfaces must use a DB2 application plan which includes all of the DBRM modules in the SSI supplied plans for TSIM7000 at IMS sites and TSCM7000 at CICS sites. If your application doesn't use any SQL of its own, you may specify the TSIM7000 or TSCM7000 plans. The job to bind the plan is found in the .JCL library supplied on the install tape (INSTJOB3).

The interface works in the following manner:

1. The using program issues a LOAD (load call) function to the subroutine to load data in main memory from the named table.

   With the LOAD function, a capability is provided to select rows to be loaded. There are no restrictions on the number of rows in any table. Additional LOAD calls may be issued to load an unlimited number of tables.

2. The using program issues a GETF (get first) call to the subroutine to search and return a row from a given table. The search values can be provided in the data I/O area itself.

   If the table had not been loaded, the first GETF issued for the table will cause the entire table to be loaded.

   The first occurrence of the selected rows is returned in the I/O area. The subroutine maintains its position in the table at the first selected row unless a GETF function is issued again for the same table.

Rows are searched in memory using a binary search method whenever possible. DB2 searches using primary "order by" fields and VSAM searches using primary key fields are performed with binary searching.

3. The using program may issue a GETN (get next) call to the subroutine to return the next occurrence of the rows previously selected during the GETF call or the program may issue a new GETF call to search for a new row from the same or a different table.

4. The using program may issue a FREE (free memory) call to the subroutine to free the memory used for a given table.

The interface arguments are setup by the user program in the following structure:

```
*
* COPYBOOK FOR THE INTERFACE ARGUMENTS TO CALL
* DB2TABLP, DB2TABLO, V2STABLP OR VSMFILEP MODULES
*
*
* APPLICATION PROGRAMS CALL DB2TABLP DB2TABL0,
* V2STABLP or VSMFILEP WITH THE FOLLOWING
* ARGUMENTS.
*
*
  01   DB2TABLP-CONTROL-AREA.
   05  DB2TABLP-RESERVED                      PIC X(8).
   05  DB2TABLP-TABLE-NAME                    PIC X(36).
   05  DB2TABLP-PROGRAM-NAME                  PIC X(8).
   05  DB2TABLP-FUNCTION-CODE                 PIC X(4).
   05  DB2TABLP-RESULT-CODE                   PIC X(8).
   05  DB2TABLP-SORT-FIELD-FLAG               PIC X(1).
   05  DB2TABLP-SQL-CODE                      PIC S9(8) COMP.
   05  FILLER                              PIC X(33).

  01   DB2TABLP-IO-AREA                       PIC X(4000).

* THE IO-AREA FOR DB2 IS 4000
* THE IO-AREA FOR VSAM CAN VARY FROM 4000-32000.

  01   DB2TABLP-SORT-AREA.
```

```
05  DB2TABLP-NO-SORT-FIELDS                    PIC 9(4) COMP.
05  DB2TABLP-FIELD-ARRAY OCCURS 1 TO 50 TIMES
    DEPENDING ON DB2TABLP-NO-SORT-FIELDS
    INDEXED BY IFLD.
    10  DB2TABLP-FIELD-NAME                    PIC X(32).
    10  DB2TABLP-ORDER-BY                      PIC X(1).
```

A description of the arguments follows:

TABLE-NAME
- Length is 36 characters.
- Must be a fully qualified name (AUTHID.TABLENAME or VSAM.TABLENAME).

PROGRAM-NAME
Name of the calling program.

FUNCTION-CODE
Length is 4 characters. See following pages.

RESULT-CODE
Length of 8 characters which is the result of the function performed.

SORT-FIELD-FLAG
= Y if during initial load of the DB2 table the rows are to be ordered. If this flag is present, then NO-OF-SORT FIELDS must be greater than zero (DB2 use only).

I/O-AREA
- Length of 4000-32,000 characters. (The maximum of any table record.)
- Contains the values passed between the user program and the subroutine.
- Values correspond to the columns in the table as laid out in the copy code.
- Values passed to the subroutine provide selection criteria for rows to be extracted.
- See section on I/O AREA RULES.

NO-SORT-FIELDS
No of columns to be used in specifying the order by clause (DB2 use only).

FIELD-NAME
From 1 to 50 DB2 table column names to be used in the order by clause.

ORDER-BY
A = Ascending
D = Descending

CALL STRUCTURE:

```
 CALL `MODULE' USING                    DB2TABLP-CONTROL-AREA
                                        DB2TABLP-IO-AREA
                                        DB2TABLP-SORT-AREA.
```

WHERE, DB2TABLP-SORT-AREA IS PRESENT, ONLY
WHEN, DB2TABLP-SORT-FIELD-FLAG=`Y'

FUNCTION CODES

LOAD Load a table into main memory. If all rows are to be selected, leave DB2TABLP-IO-AREA blank, otherwise move the desired selection criteria to this area. See I/O AREA RULES.

OK	The Load function has been processed satisfactorily.
NOTOK	Table has already been loaded or no rows found in table.
TBLINVLD	Table requested not found.

GETF Get First
The specified table is searched for the first row meeting the designated selection criteria.

OK	The Get First function has been processed satisfactorily.
NOTOK	No row found which met the selection criteria.
TBLINVLD	Table requested not found.
END	The table is empty.

NOTES:
1. NOTOK and END must be checked for a not found condition.
2. A LOAD of the entire table will be attempted if the table was not previously loaded.

GETN Get Next.

The specified table is searched for the next row which meets the selection criteria used with the preceding GETF or GETG for that table.

OK	The Get Next function has been processed satisfactorily.
NOTOK	The table to be searched has not been loaded.
END	No more rows found which satisfy the selection criteria.

FREE

Release a specified table from main memory.
After the table has been released, the associated memory becomes available again.
Cuts down on the amount of memory used and allows for flexibility to use an unlimited number of tables in your program.
All tables are automatically freed upon termination of the program. The free function does not reset the result code.

GETG

Get Greater Than.
The specified table is searched for the first row meeting the designated selection criteria, and has a sequence field with the lowest collating sequence value greater than the sequence field value specified in the selection criteria.

OK	The Get Greater Than function has been processed satisfactorily.
NOTOK	No record found which met the selection criteria or the attempt to load the table was unsuccessful.

NOTE: A LOAD of the entire table will be attempted if the table was not previously loaded.

GETL

Get Less Than.
The specified table is searched for the first record meeting the designated selection criteria, and has a sequence field with the highest collating sequence value less than the sequence field

OK	The Get Less Than function has been processed satisfactorily.
NOTOK	Not one record was found which met the selection criteria or the attempt to load the table was unsuccessful.

NOTE: A LOAD of the entire table will be attempted if the table was not previously loaded.

GETP	Get Previous. The specified table is searched for the previous record which meets the selection criteria used with the preceding GETN or GETL for that table.
OK	The Get Previous function has been processed satisfactorily.
NOTOK	The table to be searched has not been loaded.
END	No more records found which satisfy the selection criteria.

GETS	Get Sequential. The specified table is not searched for any specific record but the next record available after the current pointer is returned. The current pointer may have been established using any of the GET functions. If the current pointer is not established, the first record from the table is returned. Subsequent GETS call returns the next record and so on. This function completely ignores any search values present in the I-O area.
OK	The GET sequential function has been processed successfully.
END	There is no additional record left in the table.
TBLINVLD	The table is not preloaded.

STGF	Get Statistics data for the given table.
OK	Statistics are returned in IO-AREA (see below)
NOTOK	Table has not been loaded.

Example of a call:

```
MOVE `Authid.tblname'     TO DB2TABLP-TABLE-NAME.
MOVE `STGF'               TO DB2TABLP-FUNCTION-CODE.
CALL `MODULE' USING          DB2TABLP-CONTROL-AREA
                             DB2TABLP-IO-AREA.
IF       DB2TABLP-RESULT-CODE = `OK'
         Statistics is returned in DB2TABLP-IO-AREA
```

Statistics is returned in DB2TABLP-IO-AREA in the following format:

```
01 PRELOAD-STATISTICS.
   05  STAT-TABLENAME      PIC X(36).
   05  STAT-RECORDS        PIC 9(9) COMP.
   05  STAT-SIZE                               PIC 9(9) COMP.
   05  STAT-ACCESSES       PIC 9(9) COMP.
   05  STAT-TIMESTAMP      PIC X(8).
   05  STAT-USERID                             PIC X(8).
   05  STAT-RECLENGTH      PIC 9(4) COMP.
   05  STAT-ADDRESS **     PIC X(4).
   05  STAT-COLUMNS        PIC 9(4) COMP.
```

** STAT-ADDRESS is the address in memory where the given table has been loaded. A COBOL II program may use this address to point to a linkage section area and perform its own processing.

SUPPORT FOR EFFECTIVITY CONTROLLED TABLES - DB2 ONLY

GEFF: Get Effective Row

- This function will return a single row to the calling program from a given table and loads the table automatically in memory in sequence of its unique index columns including the Break-in date and the rows within a given combination of key values are ordered in descending values of the Break-in dates.

- The input search values in the I/O area are values for key fields where a value for the Break-in date is optional. Additionally, a value for the Break-out date may be provided.

The Break-in / Break-out values provided are used as follows:

- If break-in value is supplied, it is used; otherwise Break-in value is set to the CURRENT DATE or TIME STAMP.

- If Break-out value is supplied, it is used; otherwise Break-out effectivity check is ignored.

The effective row is selected and returned to the calling program as follows:

If Break-out date is defined in the table; then, the row selected will have a Break-in date less than or equal to the entered date and a Break-out date greater than or equal to the entered Break-in date.

If Break-out date is not defined in the table; then, the row selected will have a Break-in date less than or equal to the entered date and a computed Break-out date greater than or equal to the entered date. The Break-out date is computed as = (the Break-in date of next row - 1).

If there is no effectivity defined on the table or an SQL error is found, then a result code of `EFFERROR' is returned back to the calling program.

The following sample program shows the basics for using the interface with DB2TABLP.

```
*
  IDENTIFICATION DIVISION.
*
  PROGRAM-ID.  SAMPLE.
  AUTHOR.  SPECIALIZED SOFTWARE, INC.
  REMARKS.
*
*  THIS IS A SAMPLE BATCH PROGRAM WHICH WILL DEMONSTRATE
*  HOW TO INTERFACE WITH DB2TABLP.
*
*
  ENVIRONMENT DIVISION.
*
  CONFIGURATION SECTION.
  SOURCE-COMPUTER.  IBM-370.
  OBJECT-COMPUTER.  IBM-370.
  INPUT-OUTPUT SECTION.
  FILE CONTROL.
  SELECT IFILE ASSIGN TO UT-S-IFILE.
  SELECT OFILE ASSIGN TO UT-S-OFILE.
*
  DATA DIVISION.
*
  FILE SECTION
  FD   IFILE
       BLOCK CONTAINS 0 RECORDS
       RECORD CONTAINS 80 CHARACTERS
       RECORDING MODE IS F
       LABEL RECORDS ARE STANDARD
       DATA RECORDS ARE IFILE-RECORD
  01   IFILE-RECORD                          PIC X(80).

  FD   OFILE
       BLOCK CONTAINS 0 RECORDS
       RECORDS CONTAINS 80 CHARACTERS
       RECORDING MODE IS F
       LABEL RECORDS ARE STANDARD
       DATA RECORDS ARE OFILE-RECORDS
  01   OFILE-RECORDS                         PIC X(80)
  WORKING-STORAGE SECTION.


  01   NUMERIC-ACCUMULATORS.
   05  OT-HOURS                              PIC 9(03) VALUE ZEROS.
   05  TIME-HALF                             PIC 9(04) VALUE ZEROS.
   05  WEEK-HOURS                            PIC 9(03) VALUE ZEROS.
   05  TOTAL-HOURS                           PIC 9(03) VALUE ZEROS.

  01   PROGRAM-FLAGS.
   05  END-OF-INPUT-FILE-FLAG        PIC X(01) VALUE SPACES.
       88   END-OF-INPUT-FILE        VALUE 'Y'.
   05  ERROR-FLAG-ON                         PIC X(01) VALUE SPACES.
       88   ERROR-FLAG                       VALUE 'Y'.

  01   OUTPUT-FILE-LAYOUT.
   05  OUTPUT-EMPNO                          PIC X(04).
   05  OUTPUT-EMPAY                          PIC 9(06)V99.
   05  FILLER                        PIC X(10).
   05  OUTPUT-MESSAGE                        PIC X(25).

  01   RATE.
   05  RATE-RATENO                    PIC X(04) VALUE SPACES.
   05  RATE-FILLER01                  PIC X(01) VALUE SPACES.
   05  RATE-RATE-CODE                 PIC 9(04) VALUE ZEROS.
   05  RATE-FILLER02                  PIC X(07) VALUE SPACES.
   05  RATE-EMPLOYEE-RATE             PIC 9(04)V99 VALUE ZEROS.

  01   PAY.
   05  PAY-EMPNO                             PIC X(04).
   05  FILLER                        PIC X(01).
   05  PAY-EMPLOYEE-NAME                     PIC X(25).
   05  FILLER                        PIC X(01).
   05  PAY-WEEK-HOURS                        PIC 9(03).
   05  FILLER                        PIC X(06).
   05  PAY-RATE-CODE                         PIC 9(04).
   05  FILLER                        PIC X(07).
   05  PAY-EMPLOYEE-RATES                    PIC 9(04)V99.
   05  FILLER                        PIC X(09).
   05  PAY-TOTAL-WKPAY                       PIC 9(06)V99.
   05  FILLER                        PIC X(11).
   05  PAY-PAY-MESSAGE                       PIC X(60).
   05  FILLER                        PIC X(01).
   05  PAY-WEEKS-DATE                        PIC 9(06).
*
*LINKAGE SECTION.
*COPYBOOK TO CALL DB2TABLP OR V2STABP MODULES
*
  01   DB2TABLP-CONTROL-AREA.
   05  DB2TABLP-RESERVED                     PIC X(8).
   05  DB2TABLP-TABLE-NAME                   PIC X(36).
   05  DB2TABLP-PROGRAM-NAME         PIC X(8).
   05  DB2TABLP-FUNCTION-CODE        PIC X(4).
   05  DB2TABLP-RESULT-CODE          PIC X(8).
   05  DB2TABLP-SORT-FIELD-FLAG      PIC X(1).
   05  DB2TABLP-SQL-CODE                     PIC S9(8) COMP.
   05  FILLER                        PIC X(33).

  01   DB2TABLP-IO-AREA                      PIC X(4000).
```

```
      01    DB2TABLP-SORT-AREA.
        05  DB2TABLP-NO-SORT-FIELDS        PIC 9(4) COMP.
        05  DB2TABLP-FIELD-ARRAY OCCURS 1 TO 50 TIMES
            DEPENDING ON DB2TABLP-NO-SORT-FIELDS
            INDEXED BY IFLD.
            10  DB2TABLP-FIELD-NAME        PIC X(32).
            10  DB2TABLP-ORDER-BY          PIC X(1).
     *
     *  PROCEDURE DIVISION.
     *
        PERFORM 0010-INITIALIZATION THRU
            0010-INITIALIZATION-ZEXIT.
        PERFORM 0020-READ-INPUT-FILE THRU
            0020-READ-INPUT-FILE-ZEXIT UNTIL
            END-OF-INPUT-FILE OR
            ERROR-FLAG-ON.
        CLOSE IFILE OFILE.
        GOBACK.
      0010-INITIALIZATION.
        OPEN INPUT IFILE
            OUTPUT OFILE.
      0010-INITIALIZATION-ZEXIT.
        EXIT.
      0020-READ-INPUT-FILE.
        READ IFILE INTO PAY
            AT END MOVE 'Y' TO END-OF-INPUT-FILE-FLAG.
        IF END-OF-INPUT-FILE
          NEXT SENTENCE
        ELSE
          PERFORM 0100-READ-TABLE THRU
            0100-READ-TABLE-ZEXIT.
      0020-READ-INPUT-FILE-ZEXIT.
        EXIT.
      0100-READ-TABLE.
     *
     *  THIS ROUTINE WILL SET UP THE DB2TABLP-INTERFACE-AREA IN
     *  ORDER TO RECEIVE THE RATE TABLE INFORMATION INTO IT.
     *
        MOVE SPACES TO RATE.
        MOVE EMPNO1 TO RATE-RATENO.
        MOVE 'SPS001.RATE' TO DB2TABLP-TABLE-NAME.
```

```
  MOVE 'GETF' TO DB2TABLP-FUNCTION-CODE.
  MOVE RATE TO DB2TABLP-IO-AREA.
  MOVE 'N' TO DB2TABLP-SORT-FIELD-FLAG.
  MOVE ZERO TO DB2TABLP-NO-SORT-FIELDS.
  CALL 'DB2TABLP' USING            DB2TABLP-CONTROL-AREA
                DB2TABLP-IO-AREA.
  IF DB2TABLP-RESULT EQUAL 'NOTOK'
     DISPLAY 'RATE CODE NOT FOUND'
     DISPLAY 'DB2TABLP RESULT' DB2TABLP-RESULT
  ELSE
  IF DB2TABLP-RESULT EQUAL 'OK'
       MOVE DB2TABLP-IO-AREA TO RATE
       PERFORM 0200-COMPUTE-PAY THRU
            0200-COMPUTE-PAY-ZEXIT
  ELSE
       MOVE 'Y' TO ERROR-FLAG-ON
       DISPLAY 'ABNORMAL END OF JOB'
       DISPLAY 'DB2TABLP RESULT' DB2TABLP-RESULT.
 0100-READ-TABLE-ZEXIT.
  EXIT.
```

I/O Area Rules

The I/O area corresponds to the fields laid out in the copy code for the table as generated by DB2 DCLGEN facility or in the copy code member for the VSAM table as specified in JCL401MC.

The following search rules apply:

Blanks in the IO-AREA are interpreted as follows:

LOAD FUNCTION
All records in the given table are loaded in memory.

GETF FUNCTION
The first record loaded in memory is returned to the user program. The GETN function returns the next record in the table.

Non-blank characters in the I/O area are interpreted as follows:

LOAD FUNCTION
All records in a given table that match the given search values are loaded in main memory.

GETF FUNCTION
The first record that matches the given search values from the table loaded in memory is returned. The GETN function returns the next record that matches the search values established during the preceding GETF function. There is no need to pass any search values for the GETN function.

User VSAM File Interface

The User VSAM File Interface allows loading and processing of records from any standard VSAM KSDS. Records can be up to 32000 bytes in size and the file can contain any number of records. The documentation below describes the interfaces.

The following interfaces to VSAM files are provided:
- Batch definition facility - JCL401MC
- Batch main memory access (read only) - VSMFILEP
- CICS main memory access (read only) - V2STABLP

To use these facilities, please review the following steps:

A VSAM table can be defined in the TABLES system using the JCL member JCL401MC. This procedure examines the table record structure using its copybook and stores all relevant information in the Tables catalog. This information is then used to provide table access and maintenance.

The following should be considered when defining VSAM tables:

- Prior to running JCL401MC, the user VSAM file must be allocated and defined to VSAM using IDCAMS.

- Each VSAM file can contain one or more tables.

  If a separate VSAM file is defined for each table, the key range and record length values are not required.

  If a VSAM file contains more than one table, each table must be given its own range of key values. Additionally, if the record lengths differ, then the record length must be specified during the definition process.

• The following information is required to define a table:

Copybook member (using standard COBOL conventions)
A table name of up to 8 characters
VSAM file DD name and Dataset name
Table record length (optional)
Range of key values for the table (optional)

NOTE: The VSAM record key can be up to 255 bytes in length and it can be positioned anywhere in the first 4000 bytes of data.

For example, in the record shown below, the key starts in position 21 and is 50 bytes long.

```
START     1             21           71
          ...___________________...______________
          FLD 1         FLD 2 (KEY)        FLD 3
```

The maximum record length allowed is 32000 bytes. The key length and starting position is automatically retrieved from VSAM during the table definition.

• The maximum size table that can be loaded is 32 megabytes or the limit defined for your system for above the line memory.

To access tables defined above, it is required that the table names be qualified with a prefix of `VSAM.' For example, if the table name, as specified in JCL401MC, is RATETAB1, to access it, the name passed to the access module must be `VSAM.RATETAB1'.

• VSAM specifications for JCL401MC
JCL401MC - Extract copy code for the VSAM table. See **TABLES/AS** Reference Manual - Record Mapping for Copy Code Limitations.
  · MEMBER NAME in copy code DD may be equal to the MEMBER NAME specified in the Control DD

  · TABLE NAME specified in the Filekey Record MUST be equal to the MEMBER NAME specified in the Control DD

- DD NAME specified in the Filekey Record MUST be the DD NAME specified in the CICS FCT (File Control Table)

- CARDS 2 - 7 are continuation cards beginning in column 1 and continuing through column 80 in each card.

- DD STATEMENT in the JCL must be the same as the DD Statement in the CICS start-up JCL. The DD NAME in the Filekey Record MUST be equal to the DD NAME in this statement.

• VSAM key specifications within **TABLES/AS**
  TABLES/AS Processing Options - Database Access
  - The first time the file/table is used, the values should be checked

RECORD LENGTH	=	Maximum length for this table
KEY FIELD START	=	Location in the record where the key field begins, i.e., the 1st position = 0001
KEY FIELD LENGTH	=	Length of the key field.

The following sample program shows the basics for interfacing with VSMFILEP.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  SAMPLE.
AUTHOR. SSI.

FUNCTION: PROCESS THREE PRELOADED TABLES IN DIFFERENT WAYS
                    AS MAY BE DONE IN APPLICATIONS.

FUNCTION: THIS IS A TEST PROGRAM TO PROVIDE EXAMPLES AND
                    STATISTICS OF USING SSI'S TABLE PRELOAD
                    FACILITY AND INTERFACES WITH VS/COBOL II.

NOTES:    THE FOLLOWING IS THE BASIC PROCESS:
          -   USING THE PRELOAD FACILITY, LOAD TABLE1 AND
              TABLE3 INTO MEMORY.  THEN USING THE STAT
              FUNCTION GET THE ADDRESS OF EACH TABLE
              AND SETUP THE POINTER TO IT TO ALLOW DIRECT
              PROGRAM ACCESS.
```

- **THEN LOOP (BASED ON PARM COUNT) DOING THE FOLLOWING:**
  1. **SEQUENTIALLY PROCESS EACH ROW IN TABLE1**
  2. **FOR EACH TABLE1 ROW, SEARCH TABLE2 TO FIND THE ROW THAT MATCHES THE NON-KEY FIELD IN TABLE1. (THE FIRST GETF FUNCTION AUTOMATICALLY PRELOADS TABLE2.)**
  3. **THEN USING THE TABLE2 ROW, USE COBOLII BINARY SEARCH TO LOCATE THE ROW IN TABLE3 TO MATCH THE SECOND FIELD IN TABLE2. IF A MATCH, ADD THE TABLE2 VALUE INTO THE TABLE3 SUM VALUE.**

```
          - FINALLY, PRINT RESULTS AND COUNTS

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  IBM-370.
OBJECT-COMPUTER.  IBM-370.
INPUT-OUTPUT SECTION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  PROGRAM-LEVEL-INFO.
    05  PLI-PROGRAM          PIC X(08) VALUE 'SAMPLE '.
    05  PLI-LEVEL-LIT        PIC X(08) VALUE ' LEVEL '.
    05  PLI-LEVEL            PIX X(08) VALUE ' 1.1   '.
    05  PLI-WS-START         PIC X(25) VALUE
                            'WORKING-STORAGE-SECTION:'.

01  MISC-VALUES.
    05  WK-LOOP-COUNT        PIC 9(9) COMP.
    05  WK-COUNT-1           PIC 9(9) COMP VALUE 0.
    05  WK-COUNT-2           PIC 9(9) COMP VALUE 0.
    05  WK-COUNT-3           PIC 9(9) COMP VALUE 0.
    05  WK-COUNT-4           PIC 9(9) COMP VALUE 0.
    05  T3-SEARCH            PIC X.
        88  T3-RECORD-FOUND         VALUE 'Y'.
        88  T3-RECORD-NOT-FOUND     VALUE 'N'.

VSMFILEP INTERFACE ARGUMENTS
01  DB2TABLP-CONTROL-AREA.
    05  DB2TABLP-RESERVED           PIC X(8).
    05  DB2TABLP-TABLE-NAME         PIC X(36).
```

```
        05  DB2TABLP-PROGRAM-NAME      PIC X(8).
        05  DB2TABLP-FUNCTION-CODE     PIX X(4).
        05  DB2TABLP-RESULT-CODEPIC X(8).
        05  DB2TABLP-SORT-FIELD-FLAG   PIC X(1).
        05  DB2TABLP-SQL-CODE          PIC S9(8)  COMP.
        05  FILLER              PIC X(33).

    01  DB2TABLP-IO-AREA               PIC X(4000).

    01  DB2TABLP-SORT-AREA.
        05  DB2TABLP-NO-SORT-FIELDS    PIC 9(4)  COMP.
        05  DB2TABLP-FIELD-ARRAY OCCURS 1 TO 50 TIMES
                 INDEXED BY IFLD.
            10 DB2TABLP-FIELD-NAME     PIC X(32).
            10 DB2TABLP-ORDER-BY       PIC X(1).

    01  TABLE2    REDEFINES VSMFILEP-DATA-AREA.
        05  T2-KEY-VALUE               PIC X(03).
        05  T2-KEY-TO-T3               PIC X(01).
        05  T2-VALUE                   PIC S9(5) COMP-3.

    01  STATAREA-TABLE1.
        05  STAT-T1-TABLE-NAME         PIC X(36).
        05  STAT-T1-RECORDS            PIC 9(9) COMP.
        05  STAT-T1-SIZE               PIC 9(9) COMP.
        05  STAT-T1-ACCESSES           PIC 9(9) COMP.
        05  STAT-T1-TIMESTAMP          PIC X(8).
        05  STAT-T1-USERID             PIC X(8).
        05  STAT-T1-RECLENGTH          PIC 9(4) COMP.
        05  STAT-T1-ADDRESS            USAGE IS POINTER.
        05  STAT-T1-COLUMNS            PIC 9(4) COMP.

    01  STATAREA-TABLE2.
        05  STAT-T2-TABLE-NAME         PIC X(36).
        05  STAT-T2-RECORDS            PIC 9(9) COMP.
        05  STAT-T2-SIZE               PIC 9(9) COMP.
        05  STAT-T2-ACCESSES           PIC 9(9) COMP.
        05  STAT-T2-TIMESTAMP          PIC X(8).
        05  STAT-T2-USERID             PIC X(8).
        05  STAT-T2-RECLENGTH          PIC 9(4) COMP.
        05  STAT-T2-ADDRESS            USAGE IS POINTER.
        05  STAT-T2-COLUMNS            PIC 9(4) COMP.

    01  STATAREA-TABLE3.
        05  STAT-T3-TABLE-NAME         PIC X(36).
```

```
            05  STAT-T3-RECORDS            PIC 9(9) COMP.
            05  STAT-T3-SIZE               PIC 9(9) COMP.
            05  STAT-T3-ACCESSES           PIC 9(9) COMP.
            05  STAT-T3-TIMESTAMP          PIC X(8).
            05  STAT-T3-USERID             PIC X(8).
            05  STAT-T3-RECLENGTH          PIC 9(4) COMP.
            05  STAT-T3-ADDRESS            USAGE IS POINTER.
            05  STAT-T3-COLUMNS            PIC 9(4) COMP.




        LINKAGE SECTION.

         LS-PARMS CONTAINS THE PARM FROM THE JCL
        (PARM='00000')

        01  LS-PARMS.
            05  FILLER              PIC S9(4) COMP.
            05  LS-LOOP-COUNT              PIC 9(5).

         TABLES 1 AND 3 (SETUP USING ADDRESSES FROM TABLES
                                                  INTERFACE)

        01  TABLE1.
            05  TABLE1-RECORDS      OCCURS 1 TO 2 TIMES
                                    DEPENDING ON
                                    STAT-T1-RECORDS ASCENDING
                                    KEY IS T1-KEY-VALUE
                                    INDEXED BY T1-INDEX.
                10  T1-KEY-VALUE    PIC X(05).
                10  T1-KEY-TO-T2    PIC X(03).
        01  TABLE3.
            05  TABLE3-RECORDS      OCCURS 1 TO 2 TIMES
                                    DEPENDING ON
                                    STAT-T3-RECORDS ASCENDING
                                    KEY IS T3-KEY-VALUE
                                    INDEXED BY T3-INDEX.
                10  T3-KEY-VALUE    PIC X(01).
                10  T3-SUM          PIC S9(15) COMP-3.
        ======================================================
        =
                          PROCEDURE DIVISION
```

```
======================================================
=
PROCEDURE DIVISION    USING    LS-PARMS.

 GET THE NUMBER OF TIMES TO LOOP
                            (IF INVALID, DEFAULT = 100)

    IF LS-LOOP-COUNT NOT NUMERIC
       MOVE 100 TO WK-LOOP-COUNT
    ELSE
       MOVE LS-LOOP-COUNT TO WK-LOOP-COUNT.
 INITIALIZATION - FIRST LOAD TABLE1 AND TABLE3 AND
                  SETUP ADDRESSES TO THE LINKAGE AREA

    PERFORM  1000-LOAD-TABLES  THROUGH  1000-EXIT.

 MAIN PROCESSING LOOP - PROCESS NUMBER OF TIMES
                                                SPECIFIED

    PERFORM 2000-MAIN    THROUGH    2000-EXIT
            WK-LOOP-COUNT TIMES.

 DISPLAY RESULTS

    DISPLAY 'SAMPLE - SSI TABLES PRELOAD INTERFACE
                            TEST.'.
    DISPLAY '          PROCESS LOOP COUNT = ',
                            WK-LOOP-COUNT.
    DISPLAY '  '.
    DISPLAY ' '.
    DISPLAY 'SAMPLE - PROCESSING COUNTS:'.
    DISPLAY '============================'
    DISPLAY 'TABLE1 RECORDS PROCESSED  = ' WK-COUNT-1.
    DISPLAY 'TABLE2 VSMFILEP SEARCHES  = ' WK-COUNT-2.
    DISPLAY 'TABLE3 COBOL SEARCHES     = ' WK-COUNT-3.
    DISPLAY ' '.
    DISPLAY ' '.
    DISPLAY 'SAMPLE - TABLE 3 ROWS:'.
    DISPLAY '======================'.
    SET T3-INDEX TO 1.
    PERFORM STAT-T3-RECORDS TIMES
            DISPLAY T3-KEY-VALUE (T3-INDEX), T3-SUM
                                             (T3-INDEX)
            ADD T3-SUM (T3-INDEX) TO WK-COUNT-4
            SET T3-INDEX UP BY 1
```

```
                END-PERFORM
        DISPLAY '===================='.
        DISPLAY 'TOTAL T3-SUM = ' WK-COUNT-4.
        GOBACK.

*****************************************************
*
 INITIALIZATION - LOAD TABLES : TABLE1 AND TABLE3
 (NOTE: TABLE2 IS AUTO LOADED ON FIRST GETF)
*****************************************************
*
1000-LOAD-TABLES.

 LOAD AND SET ADDRESS FOR TABLE1

    MOVE 'VSAM.TABLE1'        TO DB2TABLP-TABLE-NAME.
    MOVE 'LOAD'          TO DB2TABLP-FUNCTION-CODE.
    MOVE SPACES          TO DB2TABLP-IO-AREA.
    CALL 'VSMFILEP'      USING DB2TABLP-CONTROL-AREA,
              DB2TABLP-IO-AREA.
    IF DB2TABLP-RESULT-CODE NOT = 'OK'
        DISPLAY 'LOAD FAILED (T1), RESULT = '
                                  DB2TABLP-RESULT-CODE
        MOVE +8 TO RETURN-CODE
        GOBACK
        END-IF.
    MOVE 'STGF'          TO DB2TABLP-FUNCTION CODE.
    CALL 'VSMFILEP'      USING DB2TABLP-CONTROL-AREA,
              DB2TABLP-IO-AREA.
    IF DB2TABLP-RESULT-CODE NOT = 'OK'
        MOVE +8 TO RETURN-CODE
        DISPLAY 'STAT FAILED (T1), RESULT = '
                                  DB2TABLP-RESULT-CODE
        GOBACK
        END-IF.
    MOVE DB2TABLP-IO-AREA TO STATAREA-TABLE1.
    SET ADDRESS OF TABLE1 TO STAT-T1-ADDRESS.

 LOAD AND SET ADDRESS FOR TABLE3

    MOVE 'VSAM.TABLE3'   TO DB2TABLP-TABLE-NAME.
    MOVE 'LOAD'          TO DB2TABLP-FUNCTION-CODE.
    MOVE SPACES          TO DB2TABLP-IO-AREA,
    CALL 'VSMFILEP'      USING DB2TABLP-CONTROL-AREA,
              DB2TABLP-IO-AREA.
```

```
           IF DB2TABLP-RESULT-CODE NOT = 'OK'
               DISPLAY 'LOAD FAILED (T3), RESULT = '
                                           DB2TABLP-RESULT-CODE
               MOVE +8 TO RETURN-CODE
               GOBACK
               END-IF.
           MOVE 'STGF'          TO DB2TABLP-FUNCTION-CODE.
           CALL 'VSMFILEP'      USING DB2TABLP-CONTROL-AREA,
                     DB2TABLP-IO-AREA.

           IF DB2TABLP-RESULT-CODE NOT = 'OK'
               DISPLAY 'STAT FAILED (T3), RESULT = '
                                           DB2TABLP-RESULT-CODE
               MOVE +8 TO RETURN-CODE
               GOBACK
               END-IF.
           MOVE DB2TABLP-IO-AREA TO STATAREA-TABLE3.
           SET ADDRESS OF TABLE3 TO STAT-T3-ADDRESS.
       1000-EXIT.
           EXIT.

      ******************************************************
      *

        MAIN PROCESSING LOOP

      ******************************************************
      *
       2000-MAIN.
           SET T1-INDEX TO 1.
           MOVE 'VSAM.TABLE2'   TO DB2TABLP-TABLE-NAME.
           MOVE 'GETF'          TO DB2TABLP-FUNCTION-CODE.
           PERFORM STAT-T1-RECORDS TIMES
              MOVE T1-KEY-TO-T2 (T1-INDEX) TO
               DB2TABLP-IO-AREA
              ADD 1 TO WK-COUNT-1
              CALL 'VSMFILEP'   USING DB2TABLP-CONTROL-AREA,
                     DB2TABLP-IO-AREA
              ADD 1 TO WK-COUNT-2
              IF DB2TABLP-RESULT-CODE = 'OK'
               PERFORM 9001-SEARCH-TABLE3 THRU 9001-EXIT
                  IF T3-RECORD-FOUND
                    ADD T2-VALUE TO T3-SUM (T3-INDEX)
                  END-IF
               END-IF
```

```
            SET T1-INDEX UP BY 1
        END-PERFORM.
    2000-EXIT.
        EXIT.
    ******************************************************
    *

     SEARCH TABLE 3 ROUTINE

    ******************************************************
    *9001-SEARCH-TABLE3.
        ADD 1 TO WK-COUNT-3.
        SET T3-RECORD-FOUND TO TRUE.
        SEARCH ALL TABLE3-RECORDS
               AT END SET T3-RECORD-NOT-FOUND TO TRUE
               WHEN T3-KEY-VALUE (T3-INDEX) = T2-KEY-TO-T3
               END SEARCH.
    9001-EXIT.
        EXIT.
```

Section 3

TABLES/AS Technical Guide

Online Maintenance

Editor Programming

TABLES/AS has a facility to automatically call a user-written program, an editor, that will be used to perform functions beyond the scope of **TABLES/AS.** Editors should be kept in the same library as other on-line application programs. The name of the editor must be specified for table processing via Processing options and/or screen switching via Switching definition.

An editor may be invoked during online table maintenance in conjunction with add, change, delete, and inquire functions. An editor may modify as well as add data to table rows. It may also be invoked during the switch from one screen to another.

When the editor is done processing, it turns control back to **TABLES/AS**. If the editor has found an error, it can highlight fields on the screen and move a message to an error message field. If it finds an abnormal condition, it can return a non-zero condition code, which signals **TABLES/AS** to return to its main menu and display a message.

If the editor is called with a AB, CB, or DB screen function, it will update the row and return to **TABLES/AS** to retrieve the next record.

TABLES/AS passes control to the editor at various steps in the processing. For example:

- After it completes its own edits, but before it displays the screen.
- Before and after it retrieves a table from a database, but before it displays the screen.
- Before and after updating a database.
- Whenever it switches from one screen to another.
- Whenever the editor mode is specified by the SCREDT control table.
- After receiving a screen and before sending a screen.

The editor can determine exactly what function is about to occur by analyzing the SCREEN-FUNCTION value that **TABLES/AS** passes to it.

SCREEN-FUNCTION

- Value passed during record processing.

RB	=	Retrieve for inquiry (before retrieval - once)
RA	=	Retrieve for inquiry (after retrieval - once per row)
UB	=	Retrieve for update (before retrieval - once)
UA	=	Retrieve for update (after retrieval -once per row)
EA	=	Edit add data prior to validation process (once per row)
EC	=	Edit change data prior to validation process (once per row)
ED	=	Edit delete data prior to validation process (once per row)

AB	=	Table add process (before add - once per row)
AA	=	Table add process (after add - once per row)
DB	=	Table delete process (before delete - once per row)
DA	=	Table delete process (after delete - once per row)
CB	=	Table change process (before change - once per row)
CA	=	Table change process (after change - once per row)
SI	=	Just after Screen Input (EDIT-ERROR-FLAG may be set to `Y' - once)
SO	=	Just before Screen Output (once)

- Values passed during screen switching.

F	=	Switch from screen (There is no data in the 2 I/O areas, only in the screen copy code area.)
T	=	Switch to screen (There is no data in the 2 I/O areas, only in the screen copy code area.)

- Value passed when editor is controlling the processing flow

Blank	=	To be processed via the editor mode (SCREDT entry)

For DB2 processing screens, a facility is provided to generate an editor shell using the JCLEMG01 (see Section 4). The following should be considered:

1. Batch JCL is used to generate a program shell given:
   . a screen name
   and . a table name

2. The editor shell includes the copy code layouts of the screen and table.

   The screen copy code must be generated prior to compiling the editor. Use JCL401M3. The 01 level in copy code must be adjusted to 04 or higher when using CICS.

3. User code may be added in procedure division before compiling and linking the editor module.

4. Provides an easy way of putting together editor modules.

Compile and Link Suggestions

- Editors called by TABLES should be COBOL II
- COBOL/VS Editors should be NORES. They may work as RES.
  - If a problem is encounted, change it to NORES. If still a problem, convert it to COBOL II.
- Programs which call TABLES should be COBOL II

Editor Parameters with TABLES/AS R3.7 and Above

COBOL II <----------------- EDITOR--------------- ----> COBOL/VS							
CICS	IMS<3.1	IMS3.1&>	PARAMETER		CICS	IMS<3.1	IMS3.1&>
31	31/24	31/24	AMODE	LINK	24	24	24
ANY	ANY	ANY	RMODE	LINK	24	24	24
REUS	REUS	REUS	REUS	LINK	ANY	ANY	ANY
RENT	RENT	RENT	RENT	LINK/COMP	--	--	--
24/31	24	24/31	DATA	COMP	--	--	--
RES	RES	RES	RES	COMP	NORES	NORES	NORES
EITHER	EITHER	EITHER	DYNAM *	COMP	(1)	(1)	(1)
YES	N/A	N/A	CICS PRECOMPILE		YES	N/A	N/A

(1) Default is NODYNAM
-- Means Do Not Specify
* ANY means TABLES does not care how the program is compiled. The choice should be made based on whether the editor calls any modules and how they will be called.

COBOL II NOTES:

- If RENT then RES is automatic
- If DYNAM then RES is automatic
- Programs compiled as RENT should be linked as RENT
- If executed above 16 Meg line (AMODE(31)), must use RENT
- PRELOADED programs should be linked as REUS

Editor Program Uses

DURING TABLE RETRIEVAL

During retrieval of rows for inquiry (Function R) and retrieval of rows for update (Function U) the editor is called before starting the retrieval process and then once for each qualified row returned.

The editor is called with one of the following user function codes:

RB = read for inquiry (just prior to table retrieval)
RA = read for inquiry (just after table retrieval)
UB = read for update (just prior to table retrieval)
UA = read for update (just after table retrieval)

The data retrieved is passed along in the RECORD-IO-AREA-NEW.

Following the RB/UB functions, after return from the editor program, the data present in the RECORD-IO-AREA-NEW is used to qualify record retrievals. This assumes that the data in the RECORD-IO-AREA-NEW was modified by the editor in some suitable way.

Following the RA/UA functions, the data present in the RECORD-IO-AREA-NEW is used to format a row on the screen. Rows are returned one at a time and the editor is called for each row.

If errors are detected by the editor, set the EDIT-ERROR-FLAG to **Y** and the Screen Processor will simply display the screen, doing no further processing.

DURING THE EDITING FUNCTION

After finishing its own editing and when no errors are found, **TABLES/AS** will call the editor to perform any additional processing. The editor is called with one of the following function codes:
EA = edit for add (prior to cross table validation process)
EC=edit for change (prior to cross table validation process)
ED = edit for delete (prior to cross table validation process)

The data to be edited lies in the following areas:

FUNCTION	**IO-AREA NEW**	**IO-AREA-OLD**
EA	Y	
EC	Y	Y
ED		Y

The editor is called once for each row on the screen being updated. The row sequence number is indicated in the MAP-OCC-NO field (in the CONTROL-AREA).

The editor may use the screen copy code area to indicate messages and fields in error. If there are edit errors, the EDIT-ERROR-FLAG in the CONTROL-AREA must be set to Y by the editor.

Upon return from the editor, the EDIT-ERROR-FLAG (in the CONTROL-AREA) is checked for a value of Y. The RETURN-CC (in the CONTROL-AREA) must be zero unless an abnormal condition is to be indicated.

After all rows on the screen have been processed through the editor, the screen is sent to the user with errors indicated or the records on the screen are processed through the **TABLES/AS** validation process.

DURING THE RECORD UPDATING PROCESS

Prior to executing the database update, the editor program is called with one of the following function codes:

AB	=	ADD FUNCTION (just <u>before</u> table is added).
AA	=	ADD FUNCTION (just <u>after</u> table is added). Data for the add lies in the RECORD-IO-AREA-NEW.
DB	=	DELETE FUNCTION (just <u>before</u> table is deleted)
DA	=	DELETE FUNCTION (just <u>after</u> table is deleted. Data for the delete lies in the RECORD-IO-AREA-OLD.
CB	=	CHANGE FUNCTION (just <u>before</u> table is changed).
CA	=	CHANGE FUNCTION (just <u>after</u> table is changed). The original data lies in RECORD-IO-AREA-OLD, and the new data lies in the RECORD-IO-AREA-NEW

The editor may perform its own internal housekeeping functions in terms of any database updating. All database PCBs are available to the editor.

After returning from the editor program, the RETURN-CC field is checked for its normal value of zero. A value greater than zero will cause the **TABLES/AS** program to terminate processing, and return to its menu screen. Only catastrophic error conditions should change the RETURN-CC. Normal edit errors are handled by setting the EDIT-ERROR-FLAG to **Y**.

When the editor is called with a function of AB, CB or DB, it will update the row. By setting the EDIT-ERROR-FLAG to **U**, **TABLES/AS** will skip its update function and move on to the next record.

DURING THE SCREEN SWITCHING PROCESS

Prior to executing the actual screen switch, the editor is called once with the screen to be switched from, and then again with the screen to be switched to.

F = Screen being switched from
T = Screen being switched to

The editor may perform its own internal housekeeping functions in terms of any database updating and retrieval from databases to initialize the output screen. For the screen switching operations, the data area that should be meaningful to the editor is the SCREEN-COPY-CODE-AREA. For switching to an MFS screen, the SCREEN-COPY-CODE-AREA should reflect the MFS format of the output message.

DURING THE SCREEN INPUT/OUTPUT PROCESS

Prior to sending the output screen, the editor is called with the screen format to be sent.

SO = Just before Screen Output

The editor may read table rows in order to provide data values on the first display of the screen.

Immediately after standard edits have been processed, the editor is called with the data entered by the user.

SI = Just after Screen Input

The editor can process based on PF keys, data values, etc. For instance, if PFK-AREA EQUAL 08, the editor can issue calls to read records from a table and build and send the next screen, thereby effecting a page down function.

WHEN THE EDITOR CONTROL MODE IS SPECIFIED

When the conditions for screen switching are not met, the SCREDT control table may be used to specify the name of an editor. If an editor name is found, it will be executed prior to sending the output. The editor may perform any processing and reset the SCREEN-COPY-CODE-AREA for output for the same or a different screen.

The function code passed to the editor is a blank character.

In this mode of processing the editor is required to formulate the contents of the screen output.

The SCREDT table may be updated by using the **TABLES/AP** processing option.

Editor Program Format

An editor can be written in any programming language and use any programming style. The only requirements are:

- The arguments passed from **TABLES/AS** to the editor are defined in the proper order. In IMS, this is accomplished with the USING statement. In CICS this accomplished with the DFHCOMMAREA.
- In IMS, the entry point name must be the same as the editor module name (<u>not</u> DLITCBL).
- In CICS, the editor must be written in command level.

IMS LINKAGE AREA

```
01 SCREEN-COPY-CODE-AREA                PIC X(3000).
01 RECORD-IO-AREA-NEW                   PIC X(4000).
01 RECORD-IO-AREA-OLD                   PIC X(4000).
01  EP-CONTROL-AREA.
    05    CA-SCREEN-FUNCTION               PIC X(4).
    05    CA-PFK-PAK-AREA                  PIC 9(2).
    05    CA-CURSOR-POSITION               PIC 9(4)COMP.
```

```
          05    CA-RETURN-CC                   PIC 9(2).
          05    CA-EDIT-ERROR-FLAG             PIC X(1).
          05    CA-RECORD-NAME                 PIC X(8).
          05    CA-QUAL-NAME                   PIC X(8).
          05    CA-MAP-OCC-NO                  PIC 9(4)COMP.
          05    CA-SCREEN-NAME                 PIC X(8).
          05    CA-IO-USERID                PIC X(8).
          05    CA-TRANCODE                    PIC X(8).
          05    CA-RSVD                        PIC X(4).
          05    CA-NULLS-FLAG                  PIC X(1).
                88   CA-NULLS-SET                       VALUE 'Y'.
          05    CA-NULLS-ARRAY-OLDPTR          USAGE IS POINTER.
          05    CA-NULLS-ARRAY-NEWPTR                   USAGE IS
     POINTER.
     01   LOGICAL-TERMINAL-PCB                 PIC X(40).
     01   MODIFIABLE-TERMINAL-PCB              PIC X(40).
```

CICS DFHCOMMAREA

```
     *    EDITOR PROGRAM COMMUNICATIONS AREA
     01   EDITOR-COMMAREA.
          05    SCREEN-COPY-CODE-AREA                   PIC X(2800).
          05    RECORD-IO-AREA-NEW             PIC X(2000).
          05    RECORD-IO-AREA-OLD             PIC X(2000).
          05    CONTROL-AREA.
          05    CA-SCREEN-FUNCTION             PIC X(4).
          05    CA-PFK-PAK-AREA                PIC 9(2).
          05    CA-CURSOR POSITION             PIC 9(4) COMP.
          05    CA-RETURN-CC                   PIC 9(2).
          05    CA-EDIT-ERROR-FLAG             PIC X(1).
          05    CA-MAP-NAME                    PIC X(8).
          05    CA-EXTENDED-AREA-FLAG          PIC X.
                88  EXTENDED-AREA                       VALUE 'Y'.
          05    CA-FILLER                      PIC X(7).
          05    CA-MAP-OCC-NO                  PIC 9(4) COMP.
          05    CA-SCREEN-NAME                 PIC X(8).
          05    CA-IO-USERID                PIC X(8).
          05    CA-EXTENSION.
                10   CAX-NEW-IO-AREA-ADDR      USAGE IS POINTER.
                10   CAX-OLD-IO-AREA-ADDR      USAGE IS POINTER.
                10   CAX-FILLER-1              PIC X(4).
                10   CAX-NULLS-AREA.
                     15   CAX-NULLS-FLAG       PIC X(1).
```

```
                    88    CAX-NULLS-SET          VALUE 'Y'.
                15    CAX-NULLS-ARRAY-OLDPTR         USAGE IS POINTER.
                15    CAX-NULLS-ARRAY-NEWPTR         USAGE IS POINTER.
            10    CAX-FILLER-2                   PIC X(87).
    *     END OF COPY
```

EDITOR ARGUMENTS

SCREEN-COPY-CODE-AREA

This area will contain the screen data in the format of the screen copy code generated by the **TABLES/AS** utility program. Maximum size is 3000 bytes in IMS and 2800 bytes in CICS.

RECORD-IO-AREA-NEW

This area contains any added or changed record in the format of the copy code as originally used through the mapping process.
The IO-AREA can contain up to 4000 bytes of data in IMS and 2000 bytes in CICS with an ability to use extended IO-AREA of 4000 bytes for CICS.

RECORD-IO-AREA-OLD

This area contains the original view of a changed or soon to be deleted record as retrieved from the database.
The IO-AREA can contain up to 4000 bytes of data in IMS and 2000 bytes in CICS with an ability to use extended IO-AREA of 4000 bytes for CICS.

CONTROL-AREA

This area contains the communication control information to or from **TABLES/AS** and editors.

SCREEN-FUNCTION

• Value passed during record processing.

RB	=	Retrieve for inquiry (before retrieval - once)
RA	=	Retrieve for inquiry (after retrieval - once per row)
UB	=	Retrieve for update (before retrieval - once)
UA	=	Retrieve for update (after retrieval -once per row)
EA	=	Edit add data prior to validation process (once per row)
EC	=	Edit change data prior to validation process (once per row)
ED	=	Edit delete data prior to validation process (once per row)
AB	=	Table add process (before add - once per row)
AA	=	Table add process (after add - once per row)
DB	=	Table delete process (before delete - once per row)
DA	=	Table delete process (after delete - once per row)
CB	=	Table change process (before change - once per row)
CA	=	Table change process (after change - once per row)
SI	=	Just after Screen Input (EDIT-ERROR-FLAG may be set to `Y' (once)
SO	=	Just before Screen Output (once)

• Values passed during screen switching.

F	=	Switch from screen (There is no data in the 2 I/O areas, only in the screen copy code area.)
T	=	Switch to screen (There is no data in the 2 I/O areas, only in the screen copy code area.)

• Value passed when editor is controlling the processing flow

Blank	=	To be processed via the editor mode (SCREDT entry)

PFK-AREA
The PF key entered is available to the editor program via this field.

CURSOR-POSITION
See **TABLES/AS** Screen Programming Considerations.

RETURN-CC
It is preset to zero prior to calling the editor. To handle catastrophic conditions, reset this field with a value greater than 0 to indicate an abnormal end. This value is almost never changed.

EDIT-ERROR-FLAG
If any errors are encountered during edit processing, the editor must set this field to a value Y to indicate the error condition. This is the standard way to tell the Screen Processor to simply display the screen because there was an error detected by the editor. If the editor executes an update and moves a **U** to this field, **TABLES/AS**, upon control will ignore its update function and retrieve the next record. If the editor sets this field to a value of **I**, the Screen Processor will initialize the next screen by retrieving the data.

MAP-NAME
Name of the map representing the table being processed.

EXT-IO-AREA-USED - CICS ONLY
Must be set to 'Y' to indicate that the editor has used the extended IO areas of 4000 bytes each. If the extended IO areas were not modified, leave this field blank.

ROW-NO (to be used only for reference)
The row number being processed. This is the relative sequence number of the row on the screen for the above table. A ROW-NO = 3 indicates that it is the third row on the screen for the given table. The ROW-NO value will be greater than 1 only when the screen is designed to display multiple rows from a given table.

By knowing the value of the ROW-NO, the editor may determine what group of fields in the SCREEN-COPY-CODE-AREA are affected for attribute setting.

SCREEN-NAME
The name of the screen being processed, switched to, or switched from.

The editor can change the value of this field to specify switching to another screen. When switching to another screen, the editor is required to formulate the contents of the SCREEN-COPY-CODE-AREA which must include the cursor position and may include the display attributes.

SIGNON-USERID
The user ID entered in the IMS/CICS signon procedure.

EXT-IO-AREA-NEW - CICS ONLY
A pointer set to the editor program which points to a 4000 bytes extended IO-AREA-NEW.

EXT-IO-AREA-OLD - CICS ONLY
A pointer set to the editor program which points to a 4000 bytes extended IO-AREA-OLD.

NULLS-ARRAY-OLDPTR
This is an address which points to an array associated with IO-AREA-OLD. There is one byte associated with each column in the record area. A high value in the byte represents a NULL column. A low value is associated with NOT NULLS.

NULLS-ARRAY-NEWPTR
This is an address which points to an array associated with IO-AREA-NEW. There is one byte associated with each column in the record area. A high value in the byte represents a NULL column. A low value is associated with NOT NULLS.

LOGICAL-TERMINAL-PCB - IMS ONLY
This is the LT-PCB for the **TABLES/AS** Screen Processor.

MODIFIABLE-TERMINAL-PCB - IMS ONLY
This can be used by the editor program to send output to other devices, for instance a printer.

IMS DB USER PCB'S - IMS ONLY

TABLES/AS can pass up to 245 user-defined PCBs, so the editor can provide IMS database access in addition to their editing capabilities. For IMS, the entry point name of the editor must be the same as the name of the editor module (not DLITCBL).

INTERRUPTING NORMAL FLOW OF SCREENS VIA EDITOR

When an editor changes the screen name in the control area to tell **TABLES/AS** that a switch to a new screen is desired, you may also enter other values to cause **TABLES/AS** to perform initial retrieval before displaying the new screen. The following field values should be set in the control area:

1. PFK-PAK-AREA to process the correct map on the target screen.
2. Value **I** in the EDIT-ERROR-FLAG.
3. Screen name to switch to in the SCREEN-NAME field.
4. Initialize screen copy code area for search values or spaces for everything.

The type of initial retrieval (R/U/None) is obtained from the Process Options - Allow Function - Initial Retrieval specification for the map to be processed in the TO screen.

If the Switching Definition Record contains a Save/Restore value of 1 thru 9, **TABLES/AS** ignores this specification when using this switching method. Therefore, if there is a $$n return set up from the screen being switched to, you will get an error telling you the $$n screen could not be found.

HOW TO COMPILE AND LINK EDITORS

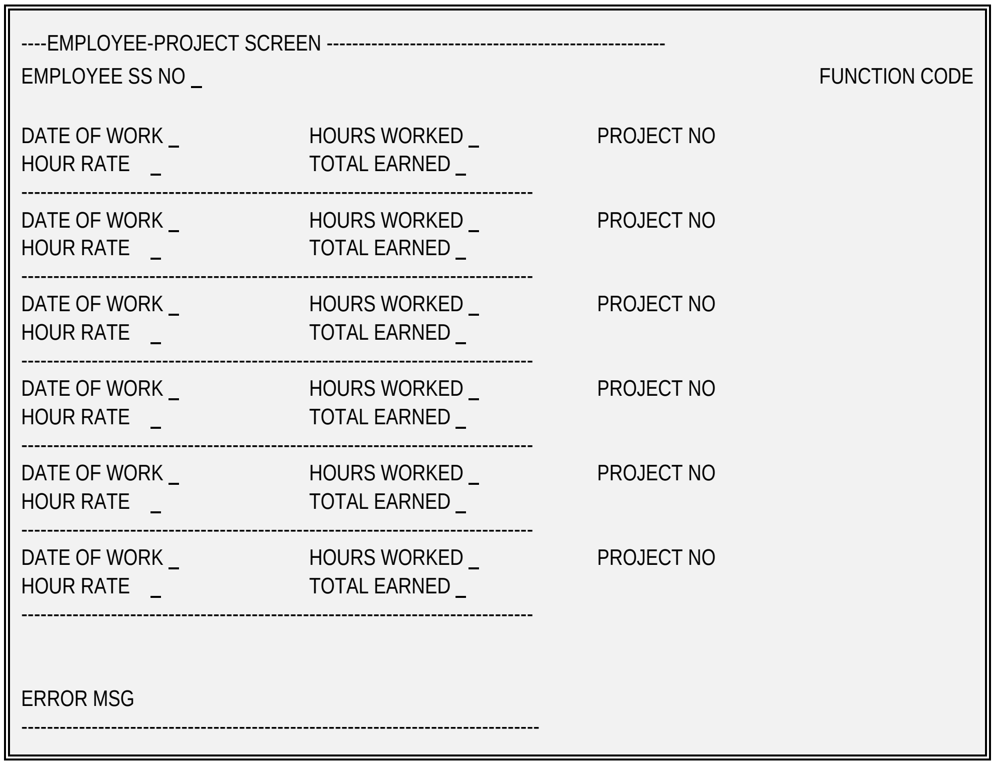
- Editors must be compiled and linked as sub-programs containing an entry point name of its own (not DLITCBL or DLITPLI in IMS).
- If the editor makes any IMS calls (CBLTDLI or PLITDLI), it must include during the linkage step the IMS control module DFSLI000 from the IMSVS.RESLIB.
- If the editor makes any DB2 call, it must include the IMS/DB2 or CICS/DB2 interface. The DBRM member must be bound in the plan used by the screen processor.
- In CICS, an entry is required in the PPT for the editor module.

EXTENDED IO AREAS FOR CICS

- Rows 2000 bytes or less do not use extended IO.
- Rows 2001 through 4000 bytes use extended IO.
  - Bytes 1-2000 are in RECORD-IO-AREA
  - Bytes 1-4000 are in EXTENDED-IO-AREA

Sample Program - Overview

Assume that a screen named EDITSCRN contains the following:



```
----EMPLOYEE-PROJECT SCREEN ----------------------------------------------------
EMPLOYEE SS NO _                                                    FUNCTION CODE

DATE OF WORK _              HOURS WORKED _              PROJECT NO
HOUR RATE    _              TOTAL EARNED _
--------------------------------------------------------------------------------
DATE OF WORK _              HOURS WORKED _              PROJECT NO
HOUR RATE    _              TOTAL EARNED _
--------------------------------------------------------------------------------
DATE OF WORK _              HOURS WORKED _              PROJECT NO
HOUR RATE    _              TOTAL EARNED _
--------------------------------------------------------------------------------
DATE OF WORK _              HOURS WORKED _              PROJECT NO
HOUR RATE    _              TOTAL EARNED _
--------------------------------------------------------------------------------
DATE OF WORK _              HOURS WORKED _              PROJECT NO
HOUR RATE    _              TOTAL EARNED _
--------------------------------------------------------------------------------
DATE OF WORK _              HOURS WORKED _              PROJECT NO
HOUR RATE    _              TOTAL EARNED _
--------------------------------------------------------------------------------


ERROR MSG
--------------------------------------------------------------------------------
```

The copy code generated for this screen is shown below. See JCL401M3 in Section 4, to generate Screen Copy Code.

```
01  EDITSCRN.
    05   EDITSCRN-EMPSSNO-ATTR-STD    PIC  X(1).
    05   EDITSCRN-EMPSSNO-ATTR-EXT    PIC  X(1).
    05   EDITSCRN-EMPSSNO-ATTR-COL    PIC  X(1).
    05   EDITSCRN-EMPSSNO-ATTR-CUR    PIC  X(1).
    05   EDITSCRN-EMPSSNO-DATA                  PIC  X(11).
    05   EDITSCRN-EDITFUNC-ATTR-STD   PIC  X(1).
    05   EDITSCRN-EDITFUNC-ATTR-EXT   PIC  X(1).
    05   EDITSCRN-EDITFUNC-ATTR-COL   PIC  X(1).
    05   EDITSCRN-EDITFUNC-ATTR-CUR   PIC  X(1).
    05   EDITSCRN-EDITFUNC-DATA                 PIC X(1).
    05   EDITSCRN-GROUP001                      OCCURS 6 TIMES.
    10 EDITSCRN-DATEWORK-ATTR-STD     PIC X(1).
         10  EDITSCRN-DATEWORK-ATTR-EXT         PIC  X(1).
         10  EDITSCRN-DATEWORK-ATTR-COL         PIC  X(1).
         10  EDITSCRN-DATEWORK-ATTR-CUR         PIC  X(1).
         10  EDITSCRN-DATEWORK-DATA   PIC  X(6).
         10  EDITSCRN-HOURS-ATTR-STD  PIC  X(1).
         10  EDITSCRN-HOURS-ATTR-EXT  PIC  X(1).
         10  EDITSCRN-HOURS-ATTR-COL  PIC  X(1).
         10  EDITSCRN-HOURS-ATTR-CUR  PIC  X(1).
         10  EDITSCRN-HOURS-DATA                PIC  9(5).
         10  EDITSCRN-PROJECT-ATTR-STDPIC  X(1).
         10  EDITSCRN-PROJECT-ATTR-EXTPIC  X(1).
         10  EDITSCRN-PROJECT-ATTR-COL          PIC  X(1).
         10  EDITSCRN-PROJECT-ATTR-CUR          PIC  X(1).
         10  EDITSCRN-PROJECT-DATA    PIC  X(6).
         10  EDITSCRN-RATE-ATTR-STD   PIC  X(1).
         10  EDITSCRN-RATE-ATTR-EXT   PIC  X(1).
         10  EDITSCRN-RATE-ATTR-COL   PIC  X(1).
         10  EDITSCRN-RATE-ATTR-CUR   PIC  X(1).
         10  EDITSCRN-RATE-DATA       PIC  9(5).
         10  EDITSCRN-EARNED-ATTR-STD PIC  X(1).
         10  EDITSCRN-EARNED-ATTR-EXT PIC  X(1).
```

```
           10  EDITSCRN-EARNED-ATTR-COLPIC  X(1).
           10  EDITSCRN-EARNED-ATTR-CURPIC  X(1).
           10  EDITSCRN-EARNED-DATA              PIC  9(6).
       05  EDITSCRN-EDITMSG-ATTR-STD    PIC  X(1).
       05  EDITSCRN-EDITMSG-ATTR-EXT    PIC  X(1).
       05  EDITSCRN-EDITMSG-ATTR-COL    PIC  X(1).
       05  EDITSCRN-EDITMSG-ATTR-CUR    PIC  X(1).
       05  EDITSCRN-EDITMSG-DATA        PIC  X(73).
```

Data entered on the screen is stored in a table called EDTTST with the following record format.

```
   01  EDTTST.
       05  EDTTST-EMPSSNO                    PIC  X(11).
       05  EDTTST-DATEWORK                   PIC S9(06).
       05  EDTTST-HOURS                      PIC S9(04)V99.
       05  EDTTST-PROJECT                    PIC  X(06).
       05  EDTTST-FILLER01                   PIC  X(01).
       05  EDTTST-RATE                       PIC S9(04)V99.
       05  EDTTST-FILLER02                   PIC  X(09).
       05  EDTTST-EARNED                     PIC S9(05)V99.
```

We want the editor to calculate the earned dollar amount for the add function and place the calculated field value in the table field EDTTST-EARNED.

To accomplish the above, the editor upon entry will check the SCREEN-FUNCTION, compute the earned dollars, and move the computed value into the EDTTST-EARNED field.

Editor Program Coding - IMS

```
 IDENTIFICATION DIVISION.

 PROGRAM-ID. LOADEXAMP.
 AUTHOR. SPECIALIZED SOFTWARE INTERNATIONAL, INC.
 DATE-COMPILED.
 REMARKS.

*  THIS IS AN EXAMPLE OF A TABLES/AS EDITOR PROGRAM.

 ENVIRONMENT DIVISION.

 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-3090.
 OBJECT-COMPUTER. IBM-3090.
 INPUT-OUTPUT SECTION.

 DATA DIVISION.

 WORKING STORAGE SECTION.

 LINKAGE SECTION.

 01    EDITSCRN.
   05  EDITSCRN-EMPSSNO-ATTR-STD           PIC  X(1).
   05  EDITSCRN-EMPSSNO-ATTR-EXT           PIC  X(1).
   05  EDITSCRN-EMPSSNO-ATTR-COL           PIC  X(1).
   05  EDITSCRN-EMPSSNO-ATTR-CUR           PIC  X(1).
   05  EDITSCRN-EMPSSNO-DATA                       PIC  X(11).
   05  EDITSCRN-EDITFUNC-ATTR-STD          PIC  X(1).
   05  EDITSCRN-EDITFUNC-ATTR-EXT          PIC  X(1).
   05  EDITSCRN-EDITFUNC-ATTR-COL          PIC  X(1).
   05  EDITSCRN-EDITFUNC-ATTR-CUR          PIC  X(1).
   05  EDITSCRN-EDITFUNC-DATA                      PIC  X(1).
   05  EDITSCRN-GROUP001                                  OCCURS 6 TIMES.
       10  EDITSCRN-DATEWORK-ATTR-STD      PIC  X(1).
       10  EDITSCRN-DATEWORK-ATTR-EXT      PIC  X(1).
       10  EDITSCRN-DATEWORK-ATTR-COL      PIC  X(1).
       10  EDITSCRN-DATEWORK-ATTR-CUR      PIC  X(1).
       10  EDITSCRN-DATEWORK-DATA          PIC  X(6).
```

```
          10  EDITSCRN-HOURS-ATTR-STD          PIC  X(1).
          10  EDITSCRN-HOURS-ATTR-EXT          PIC  X(1).
          10  EDITSCRN-HOURS-ATTR-COL          PIC  X(1).
          10  EDITSCRN-HOURS-ATTR-CUR          PIC  X(1).
          10  EDITSCRN-HOURS-DATA                    PIC  9(5).
          10  EDITSCRN-PROJECT-ATTR-STD              PIC  X(1).
          10  EDITSCRN-PROJECT-ATTR-EXT              PIC  X(1).
          10  EDITSCRN-PROJECT-ATTR-COL              PIC  X(1).
          10  EDITSCRN-PROJECT-ATTR-CUR              PIC  X(1).
          10  EDITSCRN-PROJECT-DATA        PIC  X(6).
          10  EDITSCRN-RATE-ATTR-STD       PIC  X(1).
          10  EDITSCRN-RATE-ATTR-EXT       PIC  X(1).
          10  EDITSCRN-RATE-ATTR-COL       PIC  X(1).
          10  EDITSCRN-RATE-ATTR-CUR       PIC  X(1).
          10  EDITSCRN-RATE-DATA                     PIC  9(5).
          10  EDITSCRN-EARNED-ATTR-STD               PIC  X(1).
          10  EDITSCRN-EARNED-ATTR-EXT               PIC  X(1).
          10  EDITSCRN-EARNED-ATTR-COL               PIC  X(1).
          10  EDITSCRN-EARNED-ATTR-CUR               PIC  X(1).
          10  EDITSCRN-EARNED-DATA                   PIC  9(6).
      05  EDITSCRN-EDITMSG-ATTR-STD        PIC  X(1).
      05  EDITSCRN-EDITMSG-ATTR-EXT        PIC  X(1).
      05  EDITSCRN-EDITMSG-ATTR-COL        PIC  X(1).
      05  EDITSCRN-EDITMSG-ATTR-CUR        PIC  X(1).
      05  EDITSCRN-EDITMSG-DATA                  PIC  X(73).

     01   EDTTST.
      05  EDTTST-EMPSSNO                               PIC  X(11).
      05  EDTTST-DATEWORK                              PIC S9(06).
      05  EDTTST-HOURS                                 PIC S9(04)V99.
      05  EDTTST-PROJECT                               PIC  X(06).
      05  EDTTST-FILLER01                              PIC  X(01).
      05  EDTTST-RATE                                  PIC S9(04)V99.
      05  EDTTST-FILLER02                              PIC  X(09).
      05  EDTTST-EARNED                                PIC S9(05)V99.

     01   EDTTST1.
      05  EDTTST1-EMPSSNO                              PIC  X(11).
      05  EDTTST1-DATEWORK                             PIC S9(06).
      05  EDTTST1-HOURS                                PIC S9(04)V99.
      05  EDTTST1-PROJECT                              PIC  X(06).
```

```
   05  EDTTST1-FILLER01                          PIC  X(01).
   05  EDTTST1-RATE                              PIC S9(04)V99.
   05  EDTTST1-FILLER02                          PIC  X(09).
   05  EDTTST1-EARNED                            PIC S9(05)V99.
```

Note: The second version of the table copy code may be set up to refer to the old record retrieved from the database.

```
 01    CONTROL-AREA.
   05  SCREEN-FUNCTION                           PIC X(4).
   05  PFK-PAK-AREA                              PIC 9(2).
   05  CURSOR-POSITION                           PIC 9(4) COMP.
   05  RETURN-CC                                 PIC 99.
   05  EDIT-ERROR-FLAG                           PIC X(1).
   05  MAP-NAME                                  PIC X(8).
   05  RESERVED                          PIC X(8).
   05  ROW-NO                            PIC 9(4) COMP.
   05  SCREEN-PROGRAM-NAME                       PIC X(8).
   05  SIGNON-USERID                             PIC X(8).
   05  EDITOR-TRANCODE                           PIC X(8).

 01    LOGICAL-TERMINAL-PCB.
   05  IO-TERMINAL                               PIC X(8).
   05  IO-RESERVE                                PIC X(02).
   05  IO-STATUS                                 PIC X(02).
   05  IO-PREFIX                         PIC X(12).
   05  IO-MOD-NAME                               PIC X(8).
   05  IO-USERID                         PIC X(8).

 01    MODIFIABLE-TERMINAL-PCB                   PIC X(40).
```

NOTE: To access user IMS databases, add 1 to n PCB's depending on the number of databases being accessed. These PCB's must have been included in the TSIM0900 PSB and a PSB GEN and ACB GEN must have been run.

```
PROCEDURE DIVISION USING   EDITSCRN
                           EDTTST
                           EDTTST1
                           CONTROL-AREA
```

```
                                     LOGICAL-TERMINAL-PCB
                                     MODIFIABLE-TERMINAL-PCB.

           IF SCREEN-FUNCTION = 'AB'
               COMPUTE EDTTST-EARNED =
                   EDTTST-RATE * EDTTST-HOURS
           ELSE
               NEXT SENTENCE.
           GOBACK.
```

End of IMS Sample.

Editor Program Coding - CICS

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  EDSAMPLE.
 AUTHOR. SSI.
 DATE-COMPILED.
 REMARKS.

 THIS IS AN EXAMPLE OF A TABLES/AS EDITOR.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  IBM-3090.
OBJECT-COMPUTER.  IBM-3090.
INPUT-OUTPUT SECTION.
DATA DIVISION.
WORKING-STORAGE SECTION.

 01   EDITSCRN.
  05  EDITSCRN-EMPSSNO-ATTR-STD  PIC  X(1).
  05  EDITSCRN-EMPSSNO-ATTR-EXT  PIC  X(1).
  05  EDITSCRN-EMPSSNO-ATTR-COL  PIC  X(1).
  05  EDITSCRN-EMPSSNO-ATTR-CUR  PIC  X(1).
  05  EDITSCRN-EMPSSNO-DATA              PIC  X(11).
  05  EDITSCRN-EDITFUNC-ATTR-STD  PIC  X(1).
  05  EDITSCRN-EDITFUNC-ATTR-EXT  PIC  X(1).
  05  EDITSCRN-EDITFUNC-ATTR-COL         PIC  X(1).
  05  EDITSCRN-EDITFUNC-ATTR-CUR         PIC  X(1).
  05  EDITSCRN-EDITFUNC-DATA             PIC  X(1).
  05  EDITSCRN-GROUP001                          OCCURS 6 TIMES.
      10    EDITSCRN-DATEWORK-ATTR-STD  PIC  X(1).
      10    EDITSCRN-DATEWORK-ATTR-EXT  PIC  X(1).
      10    EDITSCRN-DATEWORK-ATTR-COL PIC  X(1).
      10    EDITSCRN-DATEWORK-ATTR-CUR PIC  X(1).
      10    EDITSCRN-DATEWORK-DATA      PIC  X(6).
      10    EDITSCRN-HOURS-ATTR-STD     PIC  X(1).
      10    EDITSCRN-HOURS-ATTR-EXT     PIC  X(1).
      10    EDITSCRN-HOURS-ATTR-COL     PIC  X(1).
      10    EDITSCRN-HOURS-ATTR-CUR     PIC  X(1).
      10    EDITSCRN-HOURS-DATA         PIC  9(5).
      10    EDITSCRN-PROJECT-ATTR-STD   PIC  X(1).
```

```
         10    EDITSCRN-PROJECT-ATTR-EXT      PIC  X(1).
         10    EDITSCRN-PROJECT-ATTR-COL      PIC  X(1).
         10    EDITSCRN-PROJECT-ATTR-CUR      PIC  X(1).
         10    EDITSCRN-PROJECT-DATA          PIC  X(6).
         10    EDITSCRN-RATE-ATTR-STD  PIC  X(1).
         10    EDITSCRN-RATE-ATTR-EXT  PIC  X(1).
         10    EDITSCRN-RATE-ATTR-COL PIC  X(1).
         10    EDITSCRN-RATE-ATTR-CUR PIC  X(1).
         10    EDITSCRN-RATE-DATA             PIC  9(5).
         10    EDITSCRN-EARNED-ATTR-STD       PIC  X(1).
         10    EDITSCRN-EARNED-ATTR-EXT       PIC  X(1).
         10    EDITSCRN-EARNED-ATTR-COL       PIC  X(1).
         10    EDITSCRN-EARNED-ATTR-CUR       PIC  X(1).
         10    EDITSCRN-EARNED-DATA           PIC  9(6).
      05  EDITSCRN-EDITMSG-ATTR-STD    PIC  X(1).
      05  EDITSCRN-EDITMSG-ATTR-EXT    PIC  X(1).
      05  EDITSCRN-EDITMSG-ATTR-COL    PIC  X(1).
      05  EDITSCRN-EDITMSG-ATTR-CUR    PIC  X(1).
      05  EDITSCRN-EDITMSG-DATA               PIC  X(73).


     01   EDTTST.
      05  EDTTST-EMPSSNO                              PIC  X(11).
      05  EDTTST-DATEWORK                             PIC S9(06).
      05  EDTTST-HOURS                                PIC S9(04)V99.
      05  EDTTST-PROJECT                              PIC  X(06).
      05  EDTTST-FILLER01                             PIC  X(01).
      05  EDTTST-RATE                                 PIC S9(04)V99.
      05  EDTTST-FILLER02                             PIC  X(09).
      05  EDTTST-EARNED                               PIC S9(05)V99.


     01   EDTTST1.
      05  EDTTST1-EMPSSNO                             PIC  X(11).
      05  EDTTST1-DATEWORK                            PIC S9(06).
      05  EDTTST1-HOURS                               PIC S9(04)V99.
      05  EDTTST1-PROJECT                             PIC  X(06).
      05  EDTTST1-FILLER01                            PIC  X(01).
      05  EDTTST1-RATE                                PIC S9(04)V99.
      05  EDTTST1-FILLER02                            PIC  X(09).
      05  EDTTST1-EARNED                              PIC S9(05)V99.
```

Note: The second version of the table record copy code may be set up to refer to the old record retrieved from the data base.

```
LINKAGE SECTION.

* EDITOR PROGRAM COMMUNICATIONS AREA

01   DFHCOMMAREA.
 05  EC-SCREEN-COPY-CODE-AREA         PIC X(2800).
 05  EC-RECORD-IO-AREA-NEW            PIC X(2000).
 05  EC-RECORD-IO-AREA-OLD            PIC X(2000).
 05  EC-CONTROL-AREA.
     10   EC-SCREEN-FUNCTION          PIC X(4).
     10   EC-PFK-AREA                      PIC 99.
     10   EC-CURSOR-POSITION          PIC 9(4) COMP.
     10   EC-RETURN-CC                     PIC 99.
     10   EC-EDIT-ERROR-FLAG          PIC X.
     10   EC-MAP-NAME                      PIC X(8).
     10   EXT-IO-AREA-FLAG                 PIC X.
     10   FILLER                           PIC X(7).
     10   EC-ROW-NO                        PIC 9(4) COMP.
     10   EC-SCREEN-NAME                   PIC X(8).
     10   EC-USER-ID                       PIC X(8).
 05  EXTENDED-IO-AREAS.
     10   EXT-IO-AREA-NEW                  POINTER.
     10   EXT-IO-AREA-OLD                  POINTER.
     10   FILLER                           PIC X(100).
*  END OF COPY

PROCEDURE DIVISION.


 MOVE EC-RECORD-IO-AREA-NEW TO EDTTST.
 IF EC-SCREEN-FUNCTION = 'AB'
    COMPUTE EDTTST-EARNED = EDTTST-RATE *
        EDTTST-HOURS
        MOVE EDTTST TO EC-RECORD-IO-AREA-NEW
 ELSE
   NEXT SENTENCE.
 EXEC CICS RETURN END-EXEC.
 GOBACK.
```

Section 4

TABLES/AS Technical Guide

Batch Facilities

Index of Members

Member	Description
JCLACV01	Archive active log records
JCLADD02	Load a table
JCLCRS02	Create cross reference reports
JCLEMG01	Generate sample editor module
JCLEXT02	Extract table records
JCLLGP02	Print maintenance log
JCLMIG02	Migrate screens
JCLPRT02	Print DB2 tables
JCLSQL01	Generate static SQL module
JCL401MC	Analyze copybook for IMS SEGMENT fields (IMS)
JCL401MC	Analyze copybook for VSAM FILE fields (CICS)
JCL401MG	Extract segment SSA from DBD source (IMS ONLY)
JCL401M3	Generate screen copybook
JCL401M4	Print screen image
JCL401M7	Create screen load module (IMS)
JCL401M7	Create screen load module (CICS)

JCLACV01 - Archive Active Log Records

This utility uses DB2 plan TBLUTIL.

The archive job provides the ability to move active (current) log records to an archive dataset based on the number of days you specify in the control record. Records moved no longer exist in the active log. This job also provides the ability to specify how long records should remain on the archive log purging those older than specified, when the LOGIN DD file is input.

Log records are created by **TABLES/AS** as a result of keying 'Y' to the question 'DB2 LOGGING REQUIRED' in the Process Options Allow Functions panel and performing A/C/D to the table.

The LOGIN dataset may be input. It would contain a previous output of this job. Records on this file older than the number of days specified in COLS 1-4 of the control card will be dropped from the new LOGOUT dataset. Other records will be written from LOGIN to LOGOUT. This processing is performed prior to processing the AUDITDB.

While processing the AUDITDB, rows dated older than the number of days specified in COLS 6-9 of the control card will be written to LOGOUT and purged from the AUDITDB. Other rows will remain on the AUDITDB.

This job displays three record counts at end.

JCL Samples: JCLACV01

```
//    JOB
//* ---------------------------------------------------------------- *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ---------------------------------------------------------------- *
//*
//* JCL: JCLACV01
//*
//* DESCRIPTION: ARCHIVE TABLES/AS DB2 TABLE AUDIT DATA BASE
//*
//*
//* NOTES: OUTPUT IS NORMALLY TO TAPE. IF THE ARCHIVE WORKS, THE
//*        ACTIVE LOG FILE IS PURGED.
//*
//* ---------------------------------------------------------------- *
//*     ********************************************************
//*     ** ARCHIVE FILES                                      **
//*     **   - IF LOGIN IS NOT USED, DD CARD IS STILL REQD.  **
//*     **   - LOGOUT IS THE ARCHIVE FOR THE ACTIVE LOG      **
//*     **   - REVIEW DSNAMES, TAPE VOL=SER AND UNIT.        **
//*     **                                                    **
//*     **   - CONTROL CARD                                   **
//*     **      COL 1 - 4 = THE NUMBER OF DAYS BEFORE THE    **
//*     **                  CURRENT DATE BELOW WHICH THE     **
//*     **                  ARCHIVE LOG RECORDS ARE TO BE    **
//*     **                  DELETED. (IE. DELETE ALL LOG     **
//*     **                  RECORDS OLDER THAN NNNN DAYS).   **
//*     **      COL 6 - 9 = THE NUMBER OF DAYS BEFORE THE    **
//*     **                  CURRENT DATE BELOW WHICH THE     **
//*     **                  ACTIVE LOG RECORDS ARE TO BE     **
//*     **                  ARCHIVED. I.E., MAINTAIN THE     **
//*     **                  ACTIVE LOG RECORDS FOR 'THIS     **
//*     **                  MANY' DAYS.                       **
//*     ********************************************************
//STA    EXEC TBLDB2,
//        FUNC=ARCHIVE                ==> EXECUTE ARCHIVE FUNCTION
//*
//* OUTPUT FILE (LOGOUT)
//*
//LOGIN    DD  DUMMY,
//        DCB=(BLKSIZE=4037,LRECL=4033,RECFM=VB)
//LOGOUT   DD  DSN=SSI.TEST.ARCHIVE.AUDITDB(+1),DISP=(,CATLG,DELETE),
//        DCB=(BLKSIZE=4037,LRECL=4033,RECFM=VB),
//        VOL=SER=SSIBKP,UNIT=TAPE,LABEL=(1,SL)
//CONTROL  DD *
0090 0030
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
```

JCLACV01 - Error Messages Generated

CONDITION CODE 8

- 'XX LOG DB READ ERROR - A status code other than GB or Spaces was returned while attempting to read the AUDITDB for update.

- 'XX' LOG DB DELETE ERROR - A status code other than Spaces was returned while attempting to delete an AUDITDB row previously retrieved for update.

NOTES: 'XX' =

'BL'	Function code passed to program TAIS0503 is invalid.
'II'	DB2 SQL error -803 or -603.
'GE'	DB2 SQL code of +100, row not found searching for a unique row.
'GB'	DB2 SQL code +100, row not found reading the table sequentially.
'ER'	A DB2 SQC code other than -803, -603 or +100 was returned while attempting to read the AUDITDB.

JCLADD02 - Load a DB2 Table

This utility uses DB2 plan TBLUTIL.

It provides the ability to take a sequential dataset in the DECLGEN format for the table and insert the rows into the DB2 table. Also, you may specify a previously defined **TABLES/AS** screen to perform editing on the data being loaded. All of the field and relational editing capabilities of **TABLES/AS** can be used. The screen must have been 'PREPARED' in **TABLES/AS** in order to utilize the defined edits.

If you do not want to perform editing using **TABLES/AS** defined edits, leave out the control card in step STA. You may also remove STA and DUMMY the EDITS DD in STB if no editing is desired.

You may specify that all existing rows in the table are to be deleted prior to trying to load the rows from the extract file.

A report identifying columns which failed the edits is produced. Rows with edit errors are not added to the DB2 table. The report prints the row in character and hex and lists all columns did not pass the edits. A count of records loaded and failed is provided.

SAMPLE ERROR REPORT

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
TABLE LOAD PROGRAM OUTPUT FOLLOWS.
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

LOADING OF RECORDS IN TABLE                = SSI006.CARCL41                    DATE = 11/09/92


-------------------------------------------------------------------------------------------RECORD NO =      36, FAILED EDITING
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----0
3.....1998-03-039999-12-31
F38380FFFF6FF6FFFFFF6FF6FF
39C90C1998003003999901203 1

ERROR COLUMNS  :
   DUPLICATE RECORD

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
TABLE LOAD PROGRAM OUTPUT FOLLOWS.
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

LOADING OF RECORDS IN TABLE   = SSI006.CARCL41                             DATE  = 11/09/92
```

```
TOTAL RECORDS LOADED                        = 00000000
TOTAL RECORDS FAILED EDIT                   = 00000036
```

JCL Samples --- JCLADD02

```
//   JOB
//* ---------------------------------------------------------------- *
//*       TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ---------------------------------------------------------------- *
//*
//* JCL: JCLADD02
//*
//* DESCRIPTION: TABLES/AS DB2 TABLE LOAD. LOAD A DB2 TABLE FROM
//*              AN EXTRACT FILE. ALL OLD RECORDS MAY BE DELETED.
//*
//* NOTES: REVIEW AND CHANGE EXTRACT FILE INPUT DATASET. CHANGE
//*        CONTROL CARD TO CORRECT TABLE NAME. THE SYSTEM PARM
//*        ON THE DSN COMMAND MAY NEED TO CHANGE ALSO.
//*
//* ---------------------------------------------------------------- *
//*      ********************************************************
//*      ** EXTRACT SCREEN RECORDS TO USE FOR EDITING          **
//*      **    - CONTROL CARD                                  **
//*      **         COL 01-08 = SCREEN NAME FOR EDITS          **
//*      **         COL 10-45 = TABLE NAME BEING PROCESSED     **
//*      ********************************************************
//STA    EXEC TBLDB2,
//         FUNC=EDIT                  ==> EDIT TABLE BEFORE LOAD
//EXTRACT  DD  DSN=&EXTRACT,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(1,1)),
//         DCB=(LRECL=8000,BLKSIZE=8000,RECFM=FB)
//CONTROL  DD  *
SCR1     SSI.DB2TBLE
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
//*
//*      ********************************************************
//*      ** LOAD A DB2 TABLE                                   **
//*      **    - EXTRACT DD IS THE INPUT AND MUST BE AN        **
//*      **         EXTRACT FILE OR THE CORRECT FORMAT         **
//*      **    - CONTROL CARD                                  **
//*      **         COL 01-36 = TABLE NAME TO LOAD             **
//*    **     COL 37   = 'N' OR ANY NON-BLANK CHAR.    **
//*    **               TO SPECIFY 'NO DELETE' OF      **
//*    **               OLD RECORDS                    **
//*    **   - SYSTSIN DD CARD                          **
//*    **     CHECK SYSTEM PARM AND PLAN NAME          **
//*    ********************************************************
//STB    EXEC TBLDB2,
//        FUNC=LOAD                 ==> LOAD A DB2 TABLE
//EXTRACT  DD  DSN=SSI.EXTRACT.DATA,DISP=OLD
//**       DCB=(LRECL=4004,BLKSIZE=4008,RECFM=VB)
//EDITS    DD  DSN=&EXTRACT,DISP=(OLD,DELETE)
//CONTROL  DD  *                 -
SSI.DB2TBLE                    N
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
```

JCLADD02 - Error Messages Generated

RETURN CODE 4

- One or more rows failed the edit tests or
  One or more rows were rejected as duplicates or
  One or more rows were rejected by DB2 and returned a NOTOK while attempting to add the row.
- NO INPUT CARD FOUND - Step STB must have a control card.
- TABLE NAME CAN NOT BE BLANK - Columns 1-36 of the STB control record contains blanks.
- TABLE 'xxx--36--xx' NOT FOUND - Step STB table name does not exist in the DB2 catalog.

RETURN CODE 8

- ABNORMAL END OF DELETE FUNCTION DB2TABLEP RESULT CODE - 'xxxxxxxx' -When attempting to delete all table rows, a condition other than OK or NOTOK was encountered.

- ABNORMAL END OF ADD FUNCTION DB2TABLEP RESULT CODE - 'xxxxxxxx' -When attempting to add a table row, a condition other than OK, NOTOK or DUPLICATE was encountered.

JCLCRS02 - Table Cross Reference

This utility uses DB2 plan TBLUTIL.

The table cross reference report can provide a cross reference for a single table or for all tables processed by TABLES/AS screens. An optional parameter allows for displaying a count of the number of rows in the table.

The table name is shown, the number of rows and then a screen name which uses the table, the processing options defined for the table and the name of each column of the table used on the screen.

If the # of rows contains an *, that means the table no longer exists.

```
DATE: 02/22/90              TABLE CROSS REFERENCE REPORT                 PAGE 1

   TABLE NAME          # ROWS  SCREEN          A C D I          COLUMN NAME
-------------------------------------------------------------------------------
   SPS005.DB2TEST      9       DB2TEST Y Y Y Y          FCHAR
                                                                FDATE
                                                                FDEC
                                                                FFULL
                                                                FHALF
                                                                FSTAMP
                                                                FTIME
                                                                FVAR

   SPS001.HITLIST *    HITLIST         Y Y Y Y          DESCRIPTION
                                                                SCREENNAME

   SPS001.JSHSSSCS     9       JSHCRSDB        Y Y Y Y          BUILDING
                                                                COURSE
                                                                INSTRUCT
                                                                ROOM
                                                                TITLE
                               JSHSS003 Y Y Y Y         BUILDING
                                                                COURSE
                                                                INSTRUCT
                                                                ROOM

   SPS001.JSHSSSGR     63      DEMOS2          Y                COURSE
                                                                GRADE
                                                                NUMBER
                               JSHGRDDB        Y Y Y Y          COURSE
                                                                GRADE
                                                                NUMBER
                               JSHSS002 Y Y Y Y         COURSE
                                                                GRADE
                                                                NUMBER
                               MCIS1           Y Y Y Y          COURSE
                                                                GRADE
                                                                NUMBER


DATE: 02/22/90         TABLE CROSS REFERENCE REPORT * TOTALS *           PAGE 6
-------------------------------------------------------------------------------
                TOTAL # RECORDS READ:                       213
                TOTAL # TABLES:                              15
                TOTAL # ALL TABLE ROWS:                   1,990

                TOTAL # SCREENS:                             24
                 TOTAL # INQUIRY ONLY SCREENS:                1
                 TOTAL # ADD ONLY SCREENS:                    0
                 TOTAL # UPDATE ONLY SCREENS:                 0
                 TOTAL # ALL FUNCTION SCREENS:               23

* Table was not found in DB2 or was unavailable
```

JCL Samples --- JCLCRS02

```
//    JOB
//* ----------------------------------------------------------------- *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ----------------------------------------------------------------- *
//*
//* JCL: JCLCRS02
//*
//* DESCRIPTION: PRINT TABLES/AS SCREEN TO DB2 TABLE
//*              CROSS-REFERENCE REPORT
//*
//*
//* NOTES: - INSURE THE 'AUTHID' IS CORRECT IN STEPS 'STA' & 'STC'
//*        - 'STG' CONTAINS THE CONTROL RECORD NECESSARY FOR TAILORING
//*              THIS JOB.
//*
//*
//* ----------------------------------------------------------------- *
//*     *********************************************************
//*     ** EXTRACT TABLE SCRRCD (TABLE PROCESSING) TABLE      **
//*     *********************************************************
//STA    EXEC TBLDB2,
//         FUNC=EXTRACT               ==> EXTRACT TABLE RECORDS
//EXTRACT  DD  DSN=&&EXSCRRC1,DISP=(NEW,PASS),SPACE=(CYL,(1,1)),
//         UNIT=SYSDA,DCB=(RECFM=VB,LRECL=4004,BLKSIZE=4008)
//CONTROL  DD  *
SSI001.SCRRCD
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
//*************************************************************
//*     **************************************************************
//*     **  SORT THE RESULTING RECORDS ON SCREEN NAME & MAP NAME   **
//*     **************************************************************
//STB    EXEC PGM=SORT,REGION=2048K
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK1  DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(NEW,DELETE)
//SORTWK2  DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(NEW,DELETE)
//SORTWK3  DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(NEW,DELETE)
//SYSOUT   DD DUMMY
//SORTIN   DD DSN=&&EXSCRRC1,DISP=(OLD,DELETE)
//SORTOUT  DD DSN=&&EXSCRRC2,DISP=(NEW,PASS),SPACE=(CYL,(1,1)),
//         UNIT=SYSDA,DCB=(RECFM=VB,LRECL=636,BLKSIZE=6364)
//SYSIN    DD *
 SORT FIELDS=(5,16,CH,A),EQUALS
/*
//*     *********************************************************
//*     ** EXTRACT TABLE SCRCON (TABLE MAPPING) TABLE        **
//*     ********************************************************
//STC    EXEC TBLDB2,
//        FUNC=EXTRACT                 ==> EXTRACT TABLE RECORDS
//EXTRACT  DD  DSN=&&EXSCRCO1,DISP=(NEW,PASS),SPACE=(CYL,(5,1)),
//        UNIT=SYSDA,DCB=(RECFM=VB,LRECL=4004,BLKSIZE=4008)
//CONTROL  DD  *
SSI001.SCRCON
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
//*     ****************************************************************
//*     **  SORT THE RESULTING RECORDS ON SCREEN NAME & MAP NAME   **
//*     ****************************************************************
//STD    EXEC PGM=SORT,REGION=2048K
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK1  DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(NEW,DELETE)
//SORTWK2  DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(NEW,DELETE)
//SORTWK3  DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(NEW,DELETE)
//SYSOUT   DD DUMMY
//SORTIN   DD DSN=&&EXSCRCO1,DISP=(OLD,DELETE)
//SORTOUT  DD DSN=&&EXSCRCO2,DISP=(NEW,PASS),SPACE=(CYL,(1,1)),
//        UNIT=SYSDA,DCB=(RECFM=VB,LRECL=636,BLKSIZE=6364)
//SYSIN    DD *
 SORT FIELDS=(5,16,CH,A),EQUALS
/*
//*     ********************************************************
//*     ** MERGE SCRCON INTO SCRRCD INTO SCREEN & MAP ORDER   **
//*     ********************************************************
//STE    EXEC TBLBATCH,
//        FUNC=CROSSMGE
//EXSCRRCD DD  DSN=&&EXSCRRC2,DISP=(OLD,DELETE)
//EXSCRCON DD  DSN=&&EXSCRCO2,DISP=(OLD,DELETE)
//CROSSMGE DD  DSN=&CROSSMGE,DISP=(NEW,PASS),SPACE=(CYL,(1,1)),
//        UNIT=SYSDA,DCB=(RECFM=VB,LRECL=636,BLKSIZE=6364)
//*     ********************************************************
//*     **  SORT THE RESULTING RECORDS ON TABLE NAME         **
//*     ********************************************************
//STF    EXEC PGM=SORT,REGION=2048K
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK1  DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(NEW,DELETE)
//SORTWK2  DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(NEW,DELETE)
//SORTWK3  DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(NEW,DELETE)
//SYSOUT   DD DUMMY
//SORTIN   DD DSN=&&CROSSMGE,DISP=(OLD,DELETE)
//SORTOUT  DD DSN=&&CROSSEXT,DISP=(NEW,PASS),SPACE=(CYL,(1,1)),
//        UNIT=SYSDA,DCB=(RECFM=VB,LRECL=636,BLKSIZE=6364)
//SYSIN    DD *
 SORT FIELDS=(30,27,CH,A),EQUALS
```

```
//*      ********************************************************
//*      ** EXTRACT TABLE ROWS & INCLUDE FOR REPORTING         **
//*      **    - CONTROL CARD                                  **
//*      **         COL  1-36 = TABLE NAME OR BLANK FOR ALL    **
//*      **         COL 38-45 = 'ROWS' TO PRINT ROW COUNTS     **
//*      **    - SYSTSIN CARDS, CHECK SYSTEM PARM AND PLAN     **
//*      ********************************************************
//STG    EXEC TBLDB2,
//         FUNC=CROSSEXT                 ==> CROSS TABLE EXTRACT
//CROSSEXT DD  DSN=&&CROSSEXT,DISP=(OLD,PASS)
//CROSSRPT DD  DSN=&&CROSSRP1,DISP=(NEW,PASS),SPACE=(CYL,(1,1)),
//         UNIT=SYSDA,DCB=(RECFM=VB,LRECL=636,BLKSIZE=6364)
//CONTROL  DD  *                     ----
SSI.DB2TABLE                         ROWS
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
//************************************************************
//** SORT RECORDS ON SOURCE/TABLE NAME/SCREEN/FIELD NAME  **
//************************************************************
//STH    EXEC PGM=SORT,REGION=2048K
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK1  DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(NEW,DELETE)
//SORTWK2  DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(NEW,DELETE)
//SORTWK3  DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(NEW,DELETE)
//SYSOUT   DD DUMMY
//SORTIN   DD DSN=&&CROSSRP1,DISP=(OLD,PASS)
//SORTOUT  DD DSN=&&CROSSRPT,DISP=(NEW,PASS),SPACE=(CYL,(1,1)),
//         UNIT=SYSDA,DCB=(RECFM=VB,LRECL=636,BLKSIZE=6364)
//SYSIN    DD *
 SORT FIELDS=(30,27,CH,A,5,8,CH,A,217,30,CH,A),EQUALS
//*      ********************************************************
//*      ** PRINT TABLE CROSS REFERENCE REPORT                 **
//*      ********************************************************
//STI    EXEC TBLDB2,
//         FUNC=CROSSRPT
//EXTRACT  DD  DSN=&&CROSSRPT,DISP=(OLD,DELETE)
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
```

JCLCRS02 - Error Messages Generated

CONDITION CODE 4

- NO INPUT CARD FOUND - No control cards were entered
- TABLE NAME CANNOT BE BLANK - The first control record is blank

in columns 1-36.

- TABLE 'xx-36-xx' NOT FOUND - Table name is not in the DB2 catalog
  or the table has more than 120 columns
  or a DB2 error has occurred trying to access the
  catalog.
- TABLE 'xx-36-xx' COULD NOT BE FOUND
  Trying to retrieve the 1st
  row (GETF) from the table, a "TBINVLD' error was
  returned.
- RC='-ssss'   '   - GETF returned other than 'OK' or
  RESULT CODE = 'xxxxxx'   'TBLINVLD'

CONDITION CODE 8

- 'xx-36-xx' 'xxxxxxxx' - GETN for the table returned other than
  'OK' or 'END' - steps STA & STC
- TABLE 'xx--36--xx' - While trying to count rows, a GETF returned
  RC='SQL Code'   other than OK, END, NOTOK, TBLINVLD,
  RESULT CODE 'xxxxxx'   TBLNTFND or Spaces.
- 'xx--36--xx' 'xxxxxx' - While trying to count rows, a GETN
  returned other than OK or END.

JCLEMG01 - Generate a Sample Editor Module

This utility uses DB2 plan TBLUTIL.

This process creates a sample editor module for a specified screen and DB2 table. The communications area required and the DB2 table IO area new and old are built. A copy statement for the screen IO area is included. Before compiling the resultant editor, the screen copy code must be extracted using JCL401M3. Coding for unique application requirements should be inserted before the compile is done also.

Editor module names should be different than Screen Load module names, DBRM module names and Static SQL. module names.

The editor should be compiled with DATA(31) and linked with RMODE(ANY) and AMODE(31). For IMS shops where their IMS level does not support DATA(31), then use DATA(24).

For a more complete review of editors, see Section 3 of this manual.

NOTE: Screen copy code must be generated (JCL401M3) prior to compiling editor programs created by JCLEMG01. In CICS only, the 01 level in the screen copy code must be adjusted to 04 or higher before compiling the editor.

JCL Samples - JCLEMG01

```
//    JOB
//* ----------------------------------------------------------------- *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ----------------------------------------------------------------- *
//*
//* JCL: JCLEMG01
//*
//* DESCRIPTION: BATCH JOB TO GENERATE A SAMPLE EDITOR MODULE FOR
//*              A SPECIFIC SCREEN AND TABLE.
//*
//* NOTES: - THE OUTPUT SOURCE (SRCOUT DD) CAN BE PASSED DIRECTLY TO A
//*          STANDARD PROC THAT DOES COBOL COMPILE AND LINK.
//*        - THE INPUT SOURCE (SRCIN DD) IS THE BASIS FOR THE CREATED
//*          PROGRAM AND SHOULD NOT BE CHANGED IN ANYWAY. IT MUST BE
//*          AVAILABLE TO ANYONE CREATING A SAMPLE EDITOR.
//*
//* ----------------------------------------------------------------- *
//*     *******************************************************
//*     ** EDITOR MODULE GENERATION CARD:                    **
//*     **   - CONTROL CARD                                  **
//*     **       COL 01-08 = NAME OF PROGRAM TO CREATE       **
//*     **       COL 10-17 = SCREEN NAME (WHICH CALLS EDITOR)**
//*     **       COL 19-54 = TABLE NAME TO GENERATE FOR      **
//*     **       COL 56-63 = MAP NAME (ONLY REQUIRED IF      **
//*     **                   MORE THAN ONE MAP GOES AGAINST  **
//*     **                   THE SAME TABLE)                 **
//*     *******************************************************
//STA    EXEC TBLDB2,
//        FUNC=EDITSAMP                 ==> CREATE SAMPLE EDITOR
//SRCIN    DD  DSN=SSI.TEST.SOURCE(EDITSAMP),DISP=SHR,DCB=BUFNO=10
//SRCOUT   DD  DSN=YOUR.SOURCE.LIBRARY(TESTSTAT),DISP=OLD
//CONTROL  DD  *                                 --------
SCR206AP SCR206A  SSI006.CARCL41
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
```

JCLEMG01 - Error Messages Generated

RETURN CODE ??

```
    -        ERROR:  SCREEN, TABLE, OR MAP NOT FOUND. CHECK  -  Self explanatory
                     CONTROL CARD INPUT FOR CORRECT NAMES
                     AND IN THE CORRECT COLUMNS.
```

RETURN CODE <5

- SAMPLE EDITOR GENERATED OK, RC = 'xx' A useable editor has been created.

RETURN CODE 21

- ERROR: INVALID SELECTION FIELD: 'field name' - Name specified in control card beginning in col 1 does not exist in the DB2 table.

**** N O T E **** The sample editor requires only one control card. Additional control cards are treated as selection fields even though the editor itself will not incorporate any selection. Remove extraneous control cards and re-run.

RETURN CODE 24

- ERROR: CONTROL CARD INPUT REQUIRED - No control card was entered.

JCLEXT02 - Extract DB2 Table Data

This utility uses DB2 plan TBLUTIL.

It creates a sequential dataset containing all data records from a specified table. The resulting dataset can be used as input for non-DB2 applications, such as report writing programs.

The dataset produced is a variable length QSAM file with LRECL=4004, and BLKSIZE=4008. The maximum size DB2 row that can be extracted is 4000 bytes.

Only one table may be extracted per run.

Extracted rows may be sorted by supplying sort control cards 2-M.

A count of the number of rows extracted is provided.

JCL Samples - JCLEXT02

```
//   JOB
//* ----------------------------------------------------------------- *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ----------------------------------------------------------------- *
//*
//* JCL: JCLEXT02
//*
//* DESCRIPTION: DB2 TABLES EXTRACT. CREATES A VARIABLE LENGTH
//*             QSAM FILE FROM A DB2 TABLE.
//*
//* NOTES: CHANGE THE OUTPUT EXTRACT FILE TO YOUR REQUIREMENTS.
//*        THE FIRST CONTROL CARD WITH THE TABLE NAME IS REQUIRED.
//*        IF YOU WISH TO ORDER THE OUTPUT, SPECIFY ADDITIONAL
//*        CONTROL CARDS WITH COLUMN NAMES.
//*
//* ----------------------------------------------------------------- *
//*      *********************************************************
//*      ** EXTRACT A TABLE                                    **
//*      **    - EXTRACT DD IS THE OUTPUT FILE                 **
//*      **    - CONTROL CARD 1                                **
//*      **        COL 01-36 = DB2 TABLE NAME TO EXTRACT       **
//*      **    - CONTROL CARDS 2-M (IF ORDER-BY REQUIRED)      **
//*      **        COL 01    = BLANK                           **
//*      **        COL 02-N  = COLUMN NAME TO ORDER BY         **
//*      **        COL N+1   = BLANK OR ','                    **
//*      **        COL N+2   = 'A' (ASCENDING, THE DEFAULT)    **
//*      **                  = 'D' (DESCENDING)                **
//*      **    - SYSTSIN DD CARD - CHECK THE SYSTEM PARM       **
//*      **                   AND PLAN NAME                    **
//*      *********************************************************
//STA    EXEC TBLDB2,
//        FUNC=EXTRACT              ==> EXTRACT DB2 RECORDS
//EXTRACT  DD  DSN=&&EXTRACT,DISP=(NEW,PASS),
//        UNIT=SYSDA,SPACE=(TRK,(2,1)),
//        DCB=(LRECL=4004,BLKSIZE=4008,RECFM=VB)
//CONTROL  DD  *
SSI.DB2TBLE
 FLD1 D
 FFULL A
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
```

JCLEXT02 - Error Messages Generated

CONDITION CODE 4

- NO INPUT CARD FOUND - No control cards were entered
- TABLE NAME CANNOT BE BLANK - The first control record is blank in columns 1-36.
- TABLE 'xx-36-xx' NOT FOUND - Table name is not in the DB2 catalog
  or the table has more than 120 columns
  or a DB2 error has occurred trying to access the catalog.
- TABLE 'xx-36-xx' COULD NOT BE FOUND - Trying to retrieve the 1st
  row (GETF) from the table, a 'TBLINVLD' error was
  returned.
- RC='-ssss'   '   - GETF returned other than 'OK' or
  RESULT CODE = 'xxxxxxxx'   'TBLINVLD'

CONDITION CODE 8

- 'xx-36-xx' 'xxxxxxxx' - GETN for the table returned other than 'OK' or 'END'.

JCLLGP02 - DB2 Maintenance Log Print

This utility uses DB2 plan TBLUTIL

It prints logged changes for a given DB2 table. The specification to log DB2 maintenance is entered in TABLES/AS on the Processing Options, Allow Functions panel. The active log (table AUDITDB) or an archive log may be used as input. A maximum row size of 594 bytes can be printed depending on the column formats and lengths. Rows that are not fully shown may be printed by customized reporting using the extract file which is generated by the log print JCL. If the current processing map for the table is different than the map used when the log was created, custom reporting will be needed to generate the correct format. See custom reporting needs for AUDITDB to see what is required to reformat the LOG-TIME-STAMP field.

The name of the JCL is JCLLGP02. Multiple table names (one per card) may be specified as input. The JCL control input is as shown below. The table name must be a fully qualified name containing 36 or less characters.

CONTROL CARD

COLUMN 1-36
Table name to print

COLUMN 38-45
Log start date, MMDDYYYY

COLUMN 47-54
Log end date, MMDDYYYY

COLUMN 56-63
'ARCHIVE' to specify that the archive log file is to be used instead of the active on-line log file.

SAMPLE REPORT

```
DATE 09/01/92              LOG RECORD IN TABLE - SPS001.JSHSSSST                PAGE NO. 1

NUMBERNAME            CITY     STATE   ZIP       AVERAGE     BIRTHDAY      CHANGED

USERID - SPS001   TERMID - TSPS0119 = JSHSS001 TIME STAMP = 1991-09-01-16.17.13.283083

RECORD ADDED

00013         JOHN DOE          WORC.     MA        01608     00.0           0000000           0000000

USERID = SPS001  TERMID = TSPS0119 SCRNID = JSHSS001 TIME STAMP = 1991-09-01-16.18.28.791331

RECORD BEFORE CHANGE

00013         SANDRA JONES      WORC.     MA        01608     00.0           0000000           0000000

USERID = SPS001  TERMID = TSPS0119 SCRNID = JSHSS001 TIME STAMP = 1991-09-01.16.18.29.220582

RECORD AFTER CHANGE

00013         G. RODRIGUEZ      WORC.     MA        01608     00.0           0000000           0000000


DATE 09/01/92            LOG RECORDS IN TABLE = SPS001.JSHSSSST                 PAGE NO. 2

TOTAL RECORDS          = 3
RECORDS ADDED   = 1
RECORDS DELETED        = 0
RECORDS CHANGED        = 1
```

JCL Samples - JCLLGP02

```
//    JOB
//* ----------------------------------------------------------------- *
//*       TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ----------------------------------------------------------------- *
//*
//* JCL: JCLLGP02
//*
//* DESCRIPTION: TABLES/AS DB2 TABLE MAINTENANCE LOG PRINT.
//*
//*
//* NOTES:
//*
//*
//* ----------------------------------------------------------------- *
//*      *********************************************************
//*      ** EXTRACT DB2 LOG RECORDS                            **
//*      **    - EXTRACT CAN BE SAVED AND USED FOR CUSTOMIZED  **
//*      **      REPORTING(CONSTANT- IST 127 BYTES, VARIABLE-  **
//*      **      NEXT 3902 BYTES WHERE IST 2 BYTES IS THE      **
//*      **      LENGTH, FILLER 7922 BYTES)                    **
//*      **    - ARCHLOG DD CARD IS ONLY REQUIRED IF 'ARCHIVE' **
//*      **      IS SPECIFIED ON THE CONTROL CARD.             **
//*      **    - CONTROL CARD                                  **
//*      **         COL 01-36 = DB2 TABLE NAME TO PRINT        **
//*      **         COL 38-45 = LOG START DATE MMDDYYYY        **
//*      **         COL 47-54 = LOG END   DATE MMDDYYYY        **
//*      **         COL 56-63 = 'ARCHIVE' TO PRINT FROM THE    **
//*      **                     ARCHLOG DD CARD INSTEAD OF     **
//*      **                     FROM THE ACTIVE LOG FILE.      **
//*      *********************************************************
//STA   EXEC TBLDB2,
//         FUNC=EXTDB2                 ==> EXTRACT LOG RECORDS
//EXTRACT  DD  DSN=&&EXTRACT,DISP=(NEW,PASS),SPACE=(TRK,(5,1)),
//         DCB=(LRECL=11955,BLKSIZE=11959,RECFM=VB),UNIT=SYSDA
//TABLEOUT DD  DSN=&&TABLEOUT,DISP=(NEW,PASS),SPACE=(TRK,(1,1)),
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=FB),UNIT=SYSDA
//ARCHLOG  DD  DSN=SSI.TEST.ARCHIVE.AUDITDB,DISP=(OLD,KEEP)
//CONTROL  DD  *                       -------- -------- -------
SPS001.JSHSSSST                        01011980 12311999 ARCHIVE
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
//*      *********************************************************
//*      ** PRINT THE LOG RECORDS                              **
//*      *********************************************************
```

```
//STB    EXEC TBLBATCH,
//         FUNC=LOGPRT2
//CONTROL  DD DSN=&&TABLEOUT,DISP=(OLD,PASS)
//EXTRACT  DD DSN=&&EXTRACT,DISP=(OLD,PASS)
```

Custom Reporting Needs for AUDITDB

AUDITDB FORMAT

```
01 LOG-OUTPUT.
         02 LOG-OUTPUT1.
             03  LOG-SIZE                PIC S9(4) COMP.
             03  LOG-GROUP-DATA.
                    10      KEY-FIELD-TABLE                          PIC X(36).
                    10      LOG-TIME-STAMP                           PIC X(8).
                    10      LOG-FUNCTION                             PIC X.
                    10      LOG-USERIDPIC X(8).
                    10      LOG-TERMID                               PIC X(8).
                    10      LOG-SCREEN                               PIC X(8).
                    10      LOG-FILLER PIC X(17).
                    10      LOG-MESSAGE                              PIC X(39).
                    10      LOG-DATA.
                            49    LOG-DATA-LEN                       PIC S9(4) USAGE
                            49    LOG-DATA-TEXT                      PIC X(3900).

NOTE:  LOG-DATA-TEXT CONTAINS THE DB2 TABLE DATA

WORKING STORAGE ENTRIES TO SUPPORT CONVERSION OF
'LOG-TIME-STAMP'.
   01    DETAIL-TIME-STAMP           PIC X(26).
   01    DATECONVT                                                   PIC X(8) VALUE 'DATECONV'.
------------------------------------------------------
   DATE/TIME CONVERSION CONTROL AREA LAYOUT AND WORKAREA
------------------------------------------------------
------------------------------------------------------

   01    DATETIME-CONVERT-CONTROL.
            05      DT-INPUT-SPEC.
                            10    DT-INPUT-CODE                      PIC X.
                            10    DT-INPUT-TYPE                      PIC X.
            05      DT-OUTPUT-SPEC.
                            10    DT-OUTPUT-CODE                     PIC X.
                            10    DT-OUTPUT-TYPE                     PIC X.
            05      DT-INPUT-LEN                                     PIC S9(4) COMP VALUE ZERO.
            05      DT-OUTPUT-LEN                                    PIC X9(4) COMP VALUE ZERO.
            05      DT-RETURN-CODE                                   PIC S9(4) COMP.
            05      DT-DATACALC-CODE                                 PIC X(2) VALUE SPACES.
            05      DT-DATACALC-VALUE                                PIC S9(8) COMP-3.
            05      DT-FILLER            PIC X(13).
   01    DATETIME-CONVERT-WORK-AREA                                  PIC X(800).

MOVE 'SB' TO DT-INPUT-SPEC
MOVE 'S ' TO DT-OUTPUT-SPEC.
CALL DATECONVT USING DATETIME-CONVERT-CONTROL
                                LOG-TIME-STAMP
```

```
                                        DETAIL-TIME-STAMP
                                        DATETIME-CONVERT-WORK-AREA.

                IF DT-RETURN-CODE NOT = 0
                     PERFORM ERROR-ROUTINE
```

JCLLGP02 - Error Messages Generated

CONDITION CODE 0

```
- 'xxxxxxxx' START DATE NOT NUMERIC - Columns 38-45 are not numeric
- 'xxxxxxxx' START DATE INVALID - Not a valid year/month & day
- 'xxxxxxxx' END DATE NOT NUMERIC - Columns 47-54 not numeric
- 'xxxxxxxx' END DATE INVALID - Not a valid year/month & day
- 'SCREEN = 'ssssssss' NOT FOUND, LOG RECORD==>l(50) ... UNABLE TO EXTRACT
                        - Log records exist for the table being printed BUT;
                          The screen doesn't exist anymore
                          OR there is not map for the table
                          in the screen anymore with Add,
                          Change or Delete function.
                          l(50) are the 1st 50 chars of the log record.
```

CONDITION CODE 4

```
- TABLE NAME CAN NOT BE BLANK - Control record is blank in 1-36.
- TABLE NAME = 'xxxxxxxx' NO RECORDS FOUND - No records were found
                   for this table or meeting the selection criteria.
```

CONDITION CODE 8

```
- 'xxxxxxxx' LOG ERROR - Prog TAIS0503 returned a status
                   other than OK or END for GU function.
- ABNORMAL END 4000-DISPLAY-NEXT-RECORD - Prog TAIS0503 returned
'xxxxxxxx' LOG ERROR         a status other than OK
                                  or END for GN function.
```

JCLMIG02 - Migrate Screens and Effectivity Definitions

Migrate screens usually involves moving a screen from one environment to another. Therefore, two different DB2 plans are usually required. One in STA & STC bound to the SSI control tables where the screen and effectivity definition will be extracted from. The second in STB and STD bound to the SSI control tables where the screen and effectivity definitions will be migrated to. The DB2 plan TBLUTIL and another with a different name are used.

If there are no effectivity definitions to be migrated, steps STC and STD should not be run. Effectivity definitions apply to a table and need only be migrated once when first defined or whenever changed. They do not have to be migrated for each screen, rather once for each table.

As more and more screens are migrated, it may be practical to convert the CHGDB2TB DD statement into a cataloged dataset for STA. This dataset would contain all test table names and their corresponding production table name. Note the example in the JCL that wild card symbols are allowed to change the table names.

If step STA fails, all remaining steps are bypassed.

For each screen migrated, a picture of the screen is printed followed by counts of the various records extracted from the SSI control tables. As each screen is loaded, counts of the various records loaded to the SSI control tables are produced. The counts must be the same for a successful migration.

For each table specified for effectivity migration, a print line is generated. For each effectivity record loaded, a print line is generated for the table being loaded.

When errors are encountered, two print lines are generated for the table. They correspond to the 'EFF' and 'ORD' rows in the MS_DEFINITION table. The printout should be checked to insure the names are correct.

To complete the migration, the following need to be moved to or recreated in the new environment, if used:
- Screen load module
- Editor load modules
- Static SQL modules & DBRMs
- Bind of DB2 plan
- Refresh all preloaded tables

See Appendix K for more detail.

JCL Samples - JCLMIG02

```
//    JOB
//* ---------------------------------------------------------------- *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ---------------------------------------------------------------- *
//*
//* JCL: JCLMIG02
//*
//* DESCRIPTION: MIGRATE TABLES/AS SCREENS FROM ONE SYSTEM TO
//*              ANOTHER.
//*
//* NOTES: - THE CHGDB2TB DD CARD IN THE EXTRACT STEP CAN BE USED
//*          TO CHANGE THE NAMES OF THE TABLES YOU ARE PROCESSING.
//*          DURING TESTING, MAP TO YOUR TEST TABLES. THEN, WHEN
//*          MIGRATING TO PRODUCTION, ENTER THE TEST TABLE NAMES
//*          AND CORRESPONDING PRODUCTION TABLE NAMES. THE CHGDB2TB
//*          DD CARD IS OPTIONAL AND DOES NOT HAVE TO BE USED.
//*
//* ---------------------------------------------------------------- *
//*     *********************************************************
//*     ** EXTRACT SCREEN RECORDS FROM TEST DB2 SYSTEM       **
//*     **    - EXTRACT AND EXTRTABL DD'S ARE THE OUTPUT      **
//*     **    - CONTROL CARD                                  **
//*     **        COL 01-08 = SCREEN NAME TO EXTRACT          **
//*     **                 (MULTIPLE CARDS ALLOWED)           **
//*     **    - CHGDB2TB CARD (OPTIONAL)                      **
//*     **        COL 01-36 = OLD DB2 TABLE NAME              **
//*     **        COL 38-74 = NEW DB2 TABLE NAME              **
//*     *********************************************************
//STA    EXEC TBLDB2,
//        FUNC=MIG2EXTR                ==> TAS SCREEN EXTRACT
//EXTRACT  DD  DSN=&&EXTRACT,DISP=(NEW,PASS),
//        UNIT=SYSDA,SPACE=(TRK,(2,1),RLSE),
//        DCB=(RECFM=FB,BLKSIZE=8000,LRECL=8000)
//EXTRTABL DD  DSN=&&EXTRACT2,DISP=(NEW,PASS),
//        UNIT=SYSDA,SPACE=(TRK,(2,1),RLSE),
//        DCB=(RECFM=FB,BLKSIZE=6320,LRECL=632)
//CONTROL  DD  *
SCRNAME1
SCRNAME2
/*
//CHGDB2TB DD  *                    ¦==> TO NAME IN COL 38
SSI.TEST.TABLE1                    SSI.PROD.TABLE1
SSI.DB2A.*                                          SSI.DB2B.*
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2A)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTILA)
  END
/*
//*      *********************************************************
//*      ** RELOAD SCREEN TO PRODUCTION DB2 SYSTEM             **
//*      *********************************************************
//STB    EXEC TBLDB2,COND=(5,LE),
//         FUNC=MIG2LOAD                  ==> TAS SCREEN LOAD
//EXTRACT  DD  DSN=&&EXTRACT,DISP=(OLD,DELETE)
//EXTRTABL DD  DSN=&&EXTRACT2,DISP=(OLD,DELETE)
//SYSTSIN  DD  *
  DSN SYSTEM(DB2B)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTILB)
  END
/*
//*
//* DESCRIPTION: MIGRATE EFFECTIVITY DEFINITION FOR DB2 TABLES
//*              **** REMOVE THE NEXT TWO STEPS IF EFFECTIVITY
//*                   MIGRATION IS NOT REQUIRED **************
//*
//*
//*
//* NOTES: - THE CHGDB2TB DD CARD IN THE EXTRACT STEP CAN BE USED
//*          TO CHANGE THE NAMES OF THE TABLES YOU ARE MIGRATING.
//*
//* ----------------------------------------------------------------- *
//*      *********************************************************
//*      ** EXTRACT SCREEN RECORDS FROM TEST DB2 SYSTEM        **
//*      **    - EXTRACT DD IS THE OUTPUT                      **
//*      **    - CONTROL DD CARD                               **
//*      **         COL 01-36 = TABLE NAME TO EXTRACT          **
//*      **                  (MULTIPLE CARDS ALLOWED)          **
//*      **    - CHGDB2TB DD CARD (OPTIONAL)                   **
//*      **         COL 01-36 = OLD DB2 TABLE NAME             **
//*      **         COL 38-74 = NEW DB2 TABLE NAME             **
//*      *********************************************************
//STC    EXEC TBLDB2,COND=(5,LE,STA.ST1),
//         FUNC=EFF2EXTR                  ==> EFFECTIVITY EXTRACT
//EXTRACT  DD  DSN=&&EXTRACT,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(TRK,(2,1),RLSE),
//         DCB=(RECFM=FB,BLKSIZE=5600,LRECL=560)
//CONTROL  DD  *
SSI.TEST.TABLE1
SSI.TEST.TABLE2
/*
//CHGDB2TB DD  *                     ¦==> TO NAME IN COL 38
SSI.TEST.TABLE1                      SSI.PROD.TABLE1
SSI.DB2A.*                           SSI.DB2B.*
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2A)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTILA)
  END
/*
//STD    EXEC TBLDB2,COND=(5,LE,STA.ST1),
//         FUNC=EFF2LOAD                   ==> EFFECTIVITY LOAD
//EXTRACT  DD  DSN=&&EXTRACT,DISP=(OLD,DELETE)
//SYSTSIN  DD  *
  DSN SYSTEM(DB2B)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTILB)
  END
/*
```

JCLMIG02 - Error Messages Generated

STA - Extract Screen Information to be Migrated

CONDITION CODE 4

- NO MAP RECORDS ARE DEFINED FOR THE SCREEN - This is only a warning message. If no maps were defined for this screen then everything is fine.

CONDITION CODE 8

- E-004--SCREEN TO BE EXTRACTED NOT FOUND - A screen name specified in a control card could not be found in the SCREENDB. All remaining steps are bypassed.
- E-009--PROCESSING ABENDED -- HIGHEST STATUS CODE - 'xx' - All remaining steps are bypassed.

CONDITION CODE 16

- ABNORMAL END = 'xxxxxxxx' - While reading the SSI control tables a return code other than OK, NOTOK or END was received. All remaining steps are bypassed.

STB - Load Screen Information to a New Environment

USER ABEND CODE ie... 0675, 0720, 2000 etc.

- while attempting to write the extract records into a new environment, a critical error occurred. TBLTABND program actually issues the abend. SYSDUMP is usually required to resolve this error.

STC - Extract Effectivity Definition Information to be Migrated

CONDITION CODE 8

- One or more tables specified in the control card(s) did not have an effectivity definition row in the MS_DEFINITION table. STD will execute migrating all other table effectivity definitions found. Verify control card table names and re-run STC/STD to migrate the correct tables.

STD - Load Effectivity Definition Information to a New Environment

CONDITION CODE 8

- One or more tables failed to be added to the MS_DEFINITION table in the new environment. Insure the DB2 subsystem and plan name specified are correct and point to the SSI control tables in the new environment. Rerun STC/STD to migrate the tables reported.

JCLPRT02 - DB2 Table Print

This utility uses DB2 plan TBLUTIL

The table print procedure produces listings of rows from specified tables.

The report format prints out a description line containing the column names of the row, and beneath this it prints all of the rows from the table, double spaced. If a table has so many columns that a row cannot fit completely on one line of 132 characters, the description line and the individual rows will wrap around to the next line. A maximum record length of 608 bytes can be printed. Tables containing longer rows will be truncated to 608 bytes.

Other options allow the user to specify how many rows are to be printed, or to supply his own top heading line which will be printed out at the top of each page, along with the date and the page number. To use your own heading, place a Y in the user-supplied heading field, and place the heading on the input card following the parameter card. The heading should be centered within the 80 column input card.

The JCL needed to run this program is called JCLPRT02. You must supply the following input data for each table. Multiple tables may be specified in one run.

COLUMN 1-36 (TABLE NAME)
Name of the table from which the records will be printed. The name must be a fully qualified DB2 table name.

THE FOLLOWING PARAMETERS ARE OPTIONAL

COLUMN 37 (USER SUPPLIED HEADING)
Y If you want your own heading to be printed. If you indicate Y, the heading must be on the following card.

COLUMN 38-42 (MAXIMUM RECORDS)
Maximum number of records to be printed, right justified in the field. If blank, all records will be printed. If specified, then the field must be leading zero filled.

CONTROL CARD 2 (If Column 37 = Y)

COLUMN 1-80
Report heading

CONTROL CARDS 3-N (If ORDER-BY REQUIRED)

COLUMN 1
Blank

COLUMN 2-N
COLUMN NAME TO ORDER BY

COLUMN N+1
Blank or `,'

COLUMN N+2
A = Ascending
D = Descending

SAMPLE REPORT

```
DATE 09/07/92      ROSS-CUSTOMER I.D. FILE REPORT                  PAGE NO. 1

CUSTCODE           CUSTNAME  CUST-CAT          AVDA                CUST-CAT
EVDA

A/A                AMERICAN AIRLINES           .3500               .2000
A/A1            AMERICAN AIRLINES              .3500               .2000
A/A2            AMERICAN AIRLINES              .3000               .1750
A/A3            AMERICAN AIRLINES              .3000               .1750
A/S             A/S EMMA EDB 5/81              .0000               .0000
AAA                AMERICAN AUTO ASSOC.        .0000               .0000
AAC                AMERICAN AIRCRAFT           .0000               .0000
AAE                AIR RESOURCES A/L           .0000               .0000
```

NOTE: The maximum number of characters printed for a DB2 column is 60. Additional characters are truncated.

JCL Samples --- JCLPRT02

```
//   JOB
//* ---------------------------------------------------------------- *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ---------------------------------------------------------------- *
//*
//* JCL: JCLPRT02
//*
//* DESCRIPTION: DB2 TABLE PRINT.
//*
//*
//* NOTES:
//*
//*
//* ---------------------------------------------------------------- *
//*     ********************************************************
//*     ** PRINT DB2 TABLE RECORDS                            **
//*     **   - CONTROL CARD 1                                 **
//*     **       COL 01-36 = TABLE NAME TO PRINT              **
//*     **       COL 37    = 'Y' (HEADING CARD PRESENT)       **
//*     **       COL 38-42 = NO. OF RECORDS TO PRINT          **
//*     **   - CONTROL CARD 2 (IF COL 37=Y)                   **
//*     **       COL 01-80 = REPORT HEADING                   **
//*     **   - CONTROL CARDS 3-M (IF ORDER-BY REQUIRED)       **
//*     **       COL 01    = BLANK                            **
//*     **       COL 02-N  = COLUMN NAME TO ORDER BY          **
//*     **       COL N+1   = BLANK OR ','                     **
//*     **       COL N+2   = 'A' (ASCENDING, THE DEFAULT)     **
//*     **       COL N+2   = 'D' (DESCENDING)                 **
//*     **   - SYSTSIN DD CARD - CHECK SYSTEM PARM            **
//*     ********************************************************
//STA    EXEC TBLDB2,
//       FUNC=RCDPRNT             ==> PRINT TABLE RECORDS
//CONTROL  DD  *                  ------
SSI.DB2TEST                         Y00010
          ROSS-CUSTOMER I.D. FILE REPORT
 CUSTCODE A
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
```

JCLPRT02 - Error Messages Generated

CONDITION CODE 4

- TABLE NAME CAN NOT BE BLANK - The first control record is blank in columns 1-36.
- TABLE NOT FOUND - The table name supplied is not a table, view or alias name in SYSIBM.SYSTABLES.
- REPORT HEADING NOT FOUND - Only 1 control record was present and column 37 had a 'Y' saying a Heading Record would follow.
- NO RECORDS FOUND - The table is empty. If the program has not been changed, it could also mean that no rows were found that matched the data entered on the control record in columns 43-62. The first 20 characters of the DB2 are used in the comparison. Only an = condition satisfies the search.

CONDITION CODE 8

- PAGTRANS RESULT = xxxxxxxx - F9999, ABNORM. END-(10-READ)	While searching SYSTABLES, a result other than OK or NOTOK was was returned. SS8601S3 program.
Where xxxxxxxx =	'FNCINVLD' - the function code from DB2PRT01 to DB2TRV01 was not 'I'.
	'TBLNTFND' - table name or alias name not found in SYSTABLES. SS8601S3 program.
	'SQL ERR CODE' - SS8601S3 prog.

CONDITION CODE 10

- PAGTRANS RESULT - xxxxxxxx - F9999, ABNORM. END-(10-READ)	xxxxxxxx = 'GTH120' - the table has more than 120 columns/row. Max. supported is 120. SS8601S3 program.

CONDITION CODE 16 (Neg. SQL Error) - Open cursor or Fetch error. SS8601S3 program.

JCLSQL01 - Generate Static SQL Module

This utility uses DB2 plan TBLUTIL.

This job produces a COBOL source module for a specific **TABLES/AS** screen and DB2 table. The source output must then be run through the DB2 pre-compiler to create a DBRM and through a COBOL compile and link. The name of the load module is entered in the Process Options - Allow Functions - Static SQL Name. Therefore, JCLSQL01 is only used when the particular table is going to be processed by a **TABLES/AS** screen in Inquiry, Add, Change or Delete mode. It is NOT required or used for cross table validation, passing data between screens or for protecting fields on a screen. The name of the DBRM must be included in plan TSxM0900 (Screen Processor) or a unique application plan may be created which contains the names of the DBRMs in the Screen Processor plan and this DBRM.

The name of the Static SQL load module should be unique as a member in the loadlib list available to the region the screen will execute in, i.e., screen load module NOT = editor module name NOT = SQL load module NOT = program name.

Control card 1 specifies the source name desired, the screen name, the DB2 table name, the processing map if multiple maps exist in the screen for this table, and an indicator if the static SQL module should NOT contain an Authid for the table.

Control card 2 through N are used to specify the table columns to be used for searching rows in the DB2 table. Each column to be used is specified on a separate card. The specific DB2 operator to be used for that field is entered on the same card, after the column name and one or more intervening spaces. Some valid operators are:

Equal	=	This is the default.
Less Than	<	
Not Equal To	<> or ˆ=	
GT than or EQ to	>= or ˆ<	

JCL Samples - JCLSQL01

```
//    JOB
//* ----------------------------------------------------------------- *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ----------------------------------------------------------------- *
//*
//* JCL: JCLSQL01
//*
//* DESCRIPTION: BATCH JOB TO GENERATE A STATIC SQL COBOL SOURCE
//*              PROGRAM FOR A SPECIFIC SCREEN AND TABLE.
//*
//* NOTES: - THE OUTPUT SOURCE (SRCOUT DD) CAN BE PASSED DIRECTLY TO A
//*          STANDARD PROC THAT DOES A DB2 PRE-COMPILE AND A COBOL II
//*          COMPILE AND LINK. ONCE COMPILED, A BIND MUST BE DONE TO
//*          CONNECT THE TABLES SCREEN PROCESSOR TRANSACTION WITH THE
//*          STATIC SQL MODULE.
//*        - THE INPUT SOURCE (SRCIN DD) IS THE BASIS FOR THE CREATED
//*          PROGRAM AND SHOULD NOT BE CHANGED IN ANYWAY. IT MUST BE
//*          AVAILABLE TO ANYONE CREATING A STATIC SQL MODULE.
//*
//*        - *** IMPORTANT *** THE LOAD MODULE GENERATED FROM THE
//*          SOURCE OUTPUT OF THIS JOB MUST NOT BE THE SAME NAME
//*          AS THE NAME OF THE SCREEN.
//*
//* ----------------------------------------------------------------- *
//*      *******************************************************
//*      ** STAT SQL MODULE CARD:                             **
//*      **    - CONTROL CARD 1                               **
//*      **        COL 01-08 = NAME OF PROGRAM TO CREATE      **
//*      **        COL 10-17 = SCREEN NAME (MOD IS USED WITH) **
//*      **        COL 19-54 = TABLE NAME TO GENERATE FOR     **
//*      **        COL 56-63 = MAP NAME (ONLY REQUIRED IF     **
//*      **                    MORE THAN ONE MAP GOES AGAINST **
//*      **                    THE SAME TABLE)                **
//*      **        COL 65    = 'N' - NO AUTHID. THE SOURCE    **
//*      **                    WILL BE GEN'ED WITH NO AUTHID. **
//*      **    - CONTROL CARD 2 ... N                         **
//*      **        COL 01-72 = NAME OF A COLUMN IN THE TABLE  **
//*      **                    FOLLOWED BY A DB2 OPERATOR     **
//*      **                    (>,=,LIKE, ..). THERE SHOULD   **
//*      **                    BE AT LEAST 1 SPACE AFTER THE  **
//*      **                    COLUMN NAME (DEFAULTS TO =).   **
//*      **                    THESE ARE USED TO GENERATE THE **
//*      **                    SELECT STATEMENT.              **
//*      **                    (TO HAVE NO SELECTION, SPECIFY **
//*      **                     'NONE' AS THE FIRST FIELD).   **
//*      *******************************************************
//STA    EXEC TBLDB2,
//          FUNC=STATSQL              ==> CREATE STATIC SQL PGM
```

```
//SRCIN    DD  DSN=SSI.TEST.SOURCE(STATSQL),DISP=SHR,DCB=BUFNO=10
//SRCOUT   DD  DSN=YOUR.SOURCE.LIBRARY(TESTSTAT),DISP=SHR
//CONTROL  DD  *
TESTSTAT TESTSCRN DB2.TABLE_NAME
COL1 >=
COL2 LIKE
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
```

JCL : GENERATE STATIC SQL MODULE

JCLSQL01 - Error Messages Generated

CONDITION CODE GREATER THAN 00

- 'ERROR: SCREEN, TABLE, OR MAP NOT FOUND. CHECK AND IN THE CORRECT COLUMNS

CONDITION CODE 21

- 'ERROR: INVALID SELECTION FIELD: column name' - The column in control record 2-N is not one of the table column names found in Map Name specified in the control record 1. Check to insure that column name is spelled correctly. If so, check to insure that the Table Name and Map name specified correspond to each other for this screen.

CONDITION CODE 24

- 'ERROR: CONTROL CARD INPUT REQUIRED - No Control record found.

JCL401MC - Analyze Copy Code

This utility uses DB2 plan TBLUTIL.

It extracts, analyzes and then loads copy code into SSI control tables. To access VSAM KSDS datasets in CICS or IMS/DB segments in IMS via **TABLES/AS**, this job must be run. For CICS, this job also gathers information regarding the specific table and dataset keys and stores them in SSI control tables. For IMS segment information, a second job should be run, JCL401MG.

The following restrictions apply to the copy code format to be used for the extraction.

- A maximum of three (3) levels will be used for the extraction.
- Only the first (1st) 01 level will be extracted.
- The maximum occurs handled is 99 times.
- The field name, occurs and # of times must be on the same record.
- The maximum number of fields extracted is 120.
- All redefines are bypassed.

DB2 COLUMN SUBDEFINITION
Another useful feature of the Analyze copy code function is to provide a mechanism to subdefine one or more columns from a DB2 table where the column is actually composed of several unique fields. Once the copy code has been analyzed, screens can be developed with the subdefined fields being treated independently on the screen. See Appendix M for more details and a suggested approach.

See Mapping in the **TABLES/AS** Reference Manual to see how to use this imported copy code in conjunction with the DB2 catalog.

CICS only____

To extract KSDS dataset information in CICS, a FILEKEY control record must be present and it must begin with VSAM. Also, a DD statement must be included where the DDNAME is the same as was specified in the FILEKEY record and the DSNAME is the name of the dataset as specified in the CICS startup. For a KSDS containing multiple tables, this job would be run multiple times, once per table.

After this job is run, the results should be verified within **TABLES/AS** by selecting Option 4 - Processing Options and then Option 3 - Data Base Access. If incorrect, changes can be made in **TABLES/AS** or make sure the correct VSAM KSDS dataset was specified in the VSAM DD statement and rerun.
______________________________________**End CICS only**

JCL SAMPLES - JCL401MC (IMS)

```
//    JOB
//* ----------------------------------------------------------------- *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ----------------------------------------------------------------- *
//*
//* JCL: JCL401MC
//*
//* DESCRIPTION: ANALYZE COBOL OR PL1 COPYCODE LAYOUTS TO BE USED
//*             ONLINE WHEN DEFINING RECORD MAPPING. THE OUTPUT IS
//*             STORED IN A DB2 CONTROL TABLE.
//*
//* NOTES: THE COPYCODE DD STATEMENT IS THE INPUT COPYBOOK FILE.
//*
//* ----------------------------------------------------------------- *
//*     **********************************************************
//*     ** ANALYZE COPYCODE                                     **
//*     **   - SET FUNCTION AS FOLLOWS:                         **
//*     **       FUNC = CPYEXTCT   FOR COBOL COPY BOOK          **
//*     **       FUNC = PLICOPY2   FOR PLI COPY BOOK            **
//*     **   - CONTROL CARD                                     **
//*     **       COL 01-08 = COPYCODE MEMBER NAME               **
//*     **********************************************************
//STA    EXEC TBLDB2,
//        FUNC=CPYEXTCT              ==> ANALYZE COBOL COPY BOOK
//COPYCODE DD  DISP=SHR,DSN=SSI.TEST.COPYLIB(SEGNAME)
//CONTROL  DD  *
SEGNAME
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
```

JCL Samples - JCL401MC (CICS)

```
//    JOB
//* ----------------------------------------------------------------- *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ----------------------------------------------------------------- *
//*
//* JCL: JCL401MC
//*
//* DESCRIPTION: ANALYZE COBOL, ASM OR PL1 COPYCODE LAYOUTS TO BE USED
//*            ONLINE WHEN DEFINING RECORD MAPPING. THE OUTPUT IS
//*            STORED IN A TABLES/AS DATABASE.
//*
//* NOTES: - THE COPYCODE DD CARD IN STEP STA IS THE INPUT COPYBOOK.
//*         (NOTE: FOR ASSEMBLER COPY BOOK, IT USES 'SYSIN' DD FOR
//*          COPY BOOK INSTEAD OF 'COPYCODE' DD.)
//*       - FOR USER VSAM FILES, SEE 'FILEKEY' DESCRIPTION BELOW.
//*
//* ----------------------------------------------------------------- *
//*     **********************************************************
//*     ** ANALYZE COPYCODE                                    **
//*     **   - SET FUNCTION AS FOLLOWS:                        **
//*     **        FUNC = CPYEXTCT  FOR COBOL COPY BOOK         **
//*     **              REQUIRES 'COPYCODE' DD                 **
//*     **        FUNC = ASMEXTCT  FOR ASM COPY BOOK           **
//*     **              REQUIRES 'SYSIN' DD FOR COPY BOOK      **
//*     **              INSTEAD OF 'COPYCODE' DD               **
//*     **        FUNC = PLICOPY2  FOR PLI COPY BOOK           **
//*     **              REQUIRES 'COPYCODE' DD                 **
//*     **   - CONTROL CARD                                    **
//*     **        COL 01-08 = TABLNAME AS TABLES/AS WILL       **
//*     **              KNOW IT.  SAME AS NAME IN FILEKEY      **
//*          **              STATEMENT                       **
//*     **********************************************************
//STA    EXEC TBLDB2,
//        FUNC=CPYEXTCT                ==> ANALYZE COBOL COPY BOOK
//SYSPUNCH DD  SYSOUT=*                                  00009200
//SYSLIB   DD  DSN=SYS1.MACLIB,DISP=SHR                  00009200
//SYSLIN   DD  SPACE=(TRK,(1,1)),UNIT=VIO                00009200
//SYSUT1   DD  SPACE=(TRK,(1,1)),UNIT=VIO                00009200
//SYSIN    DD  DISP=SHR,DSN=SSI.CICS.COPYLIB(MEMBER)     00009400
//COPYCODE DD  DISP=SHR,DSN=SSI.CICS.COPYLIB(MEMBER)
//CONTROL  DD  *
TABLNAME
//*     *************************************************************
//*     ** FOR VSAM TABLES, ENTER OPTIONAL INFORMATION:            **
//*     **  CARD 1: COL  1-4  = 'VSAM' FOR VSAM FILE              **
//*     **         COL  6-13 = TABLE NAME                         **
//*     **         COL 15-22 = DD NAME OF THE VSAM FILE           **
//*     **                   (DEFAULTS TO THE TABLE NAME)         **
//*     **         COL 24-28 = RECORD LEN (OPTIONAL, MAX=32000) **
//*     **         COL 29-80 = TABLE KEY RANGE, START RANGE       **
//*     **                   FOLLOWED BY THE END RANGE WITH       **
//*      **                    NO SPACE BETWEEN THE TWO VALUES  **
//*      **                    (IF FILE CONTAINS A SINGLE TABLE,**
//*      **                    ENTER LOW & HIGH VALUES FOR      **
//*      **                    RANGE OR LEAVE THEM BLANK)       **
//*      **  CARD 2 THRU 7 = ADDITIONAL CARDS TO CONTINUE THE     **
//*      **                 KEY FIELDS AS REQUIRED.             **
//*      ************************************************************
//FILEKEY   DD  *      -    -
VSAM TABLNAME DDNAME
/*
//*      ************************************************************
//*      **    USER VSAM FILE - DDNAME MUST MATCH THE DD NAME     **
//*      **    SPECIFIED UNDER THE CONTROL CARD FILEKEY           **
//*      ************************************************************
//DDNAME    DD DSN=USER.VSAM.TABLES,DISP=SHR
//*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2X)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
```

JCL401MC - Extract copy code for the VSAM table. See **TABLES/AS** Reference Manual - Record Mapping for copy code Limitations.

- MEMBER NAME in COPYCODE DD may be equal to the TABLNAME specified in the Control DD
- TABLNAME specified in the Filekey Record MUST be equal to TABLNAME specified in the Control DD
- DD NAME specified in the Filekey Record MUST be the DD NAME specified in the CICS FCT (File Control Table)
- CARDS 2 - 7 are continuation cards beginning in column 1 and continuing through column 80 in each card.
- DDNAME DD statement in the JCL must be the same as the DD Statement in the CICS start-up JCL. DDNAME in the Filekey Record MUST be the same as the DDNAME in this statement.

JCL401MC - Error Messages Generated

CONDITION CODE ??

- 'COPY CODE MEMBER CARD NOT FOUND' - There was no input record following the CONTROL DD * statement.
- 'NO INPUT RECORD FOUND IN CODE FILE' - The copy code member is empty. Correct the member and rerun.
- 'NUMBER OF FIELDS IN THE COPY CODE EXCEEDS 120' - Check the copy code. If there are more than 120 fields required for the extract, TABLES/AS will not be able to support this table.
- 'COPY CODE MEMBER CONTAINS UNRECOGNIZABLE STRING' - There are a number of other displays associated with this error to assist in determining the copy code record which can not be deciphered. Delete it if possible. Restate it another way if possible. Contact SSI if all else fails.
- 'UNACCEPTABLE PICTURE STRING' - There are a number of other displays associated with this error to assist in determining the copy code record which can not be deciphered. Delete it if possible. Restate it another way if possible. Contact SSI if all else fails.

CONDITION CODE 08

- 'ERROR OPENING VSAM DD NAME = xxxxxxxx FILE STATUS = xx - An error was encountered opening the VSAM dataset. Use the file status code to determine the cause and correct.
- 'DELETE ERROR ON VSMSE2 TABLE xxxxxxxx' - While attempting to delete rows to the VSMSE2 table, an abnormal condition was encountered. Contact SSI.
- 'ADD ERROR ON VSMSE2 TABLE xxxxxxxx" - While attempting to add rows to the VSMSE2 table, an abnormal condition was encountered. Contact SSI.

- 'FILE RECORD LENGTH MUST BE NUMERIC' - Positions 24-28 of the FILEKEY record were not numeric. Fix & resubmit.

ABEND U2000

- '100-MAINLINE LAYOUT(DLET)' - An unexpected error was encountered trying to delete the old extracted copy code from the SSI LAYOUT table. Contact SSI.
- '210-READ-05-CARD LAYOUT(ADD)' - An unexpected error was encountered trying to add an extracted record to the SSI LAYOUT table. Contact SSI.
- '220-READ-10-CARD LAYOUT(ADD)' - An unexpected error was encountered trying to add an extracted record to the SSI LAYOUT table. Contact SSI.

JCL401MG - Extract IMS/DB Segment Information

This utility uses DB2 plan TBLUTIL.

It extracts, analyzes and then loads segment data into SSI control tables. A DBD is analyzed to create SSA information, segment lengths and keys or sequence fields for the segments in the IMS database.

After this job is run, the results should be verified within TABLES/AS by selecting Option 4 - Processing Options and then Option 3 - Data Base Access. If incorrect, changes can be made in TABLES or review the DBD, correct it, and then rerun this job.

JCL Samples - JCL401MG (IMS only)

```
//    JOB
//* ----------------------------------------------------------------- *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ----------------------------------------------------------------- *
//*
//* JCL: JCL401MG
//*
//* DESCRIPTION: EXTRACT IMS DATABASE SEGMENT INFORMATION FROM
//*              DBD SOURCE. OUTPUT IS STORED IN A DB2 CONTROL TABLE.
//*
//* NOTES: THE DBDCODE DD CARD SHOULD POINT TO THE DBD SOURCE.
//*
//* ----------------------------------------------------------------- *
//*     ********************************************************
//*     ** ANALYZE DBD                                        **
//*     ********************************************************
//STB    EXEC TBLDB2,
//        FUNC=DBDEXTCT                ==> EXTRACT SSA FROM DBD
//DBDCODE  DD  DISP=SHR,DSN=SSI.TEST.DBDLIB(DBDNAME)
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
```

JCL401MG - Error Messages Generated

CONDITION CODE 00

- 'NO INPUT RECORD FOUND IN CODE FILE' - The DBD member is empty.
  Correct the member and rerun.
- 'CODE - dbdname'
  'DBD CODE CONTAINS UNRECOGNIZABLE STRING' - Review the DBD coding
  and insure it is all valid. Correct any errors found and
  retry. Contact SSI if all else fails.

ABEND 72000

- '7000-DELETE-CURRENT-IMSSE2(DLET)'
  An unexpected error was encountered trying to
  delete old segment information from IMSSE2 table.
  Contact SSI.
- '8200-ADD_TABLES-DLI(ADD)'
  An unexpected error was encountered trying to
  add new segment information in IMSSE2 table.
  Contact SSI.
- '9000-READ-AN-XDFLD(GETF)'
  An unexpected error was encountered trying to
  Retrieve XDFLD information in IMSSE2 table.
  Contact SSI.
- '9000-MOVE-FIELD-LENGTH(CHNG)'
  An unexpected error was encountered trying to
  change field length for XDFLD in IMSSE2 table.
  Contact SSI.

JCL401M3 - Generate Screen Copybook

This utility uses DB2 plan TBLUTIL.

This job produces a COBOL or PLI copybook member for a specific **TABLES/AS** screen. This copybook member can then be used in Editor modules or in Application programs. Multiple copybooks can be generated by supplying multiple control cards.

Only input/output fields are included in the copybook members. For each fields on the screen, five (5) fields are defined in the copybook member.

'SCREENNAME-FIELDNAME-ATTR-STD' - To change Standard Attribute.
'SCREENNAME-FIELDNAME-ATTR-EXT' - To change Extended Attribute.
'SCREENNAME-FIELDNAME-ATTR-COL' - To change the Color Attribute.
'SCREENNAME-FIELDNAME-ATTR-CUR' - To specify Cursor Position.
'SCREENNAME-FIELDNAME-DATA' - The I/O field itself.

NOTE: Screen copy code must be generated prior to compiling editor programs created by JCLEMG01. In CICS only, the 01 level in the screen copy code must be adjusted to 04 or higher before compiling the editor.

SAMPLE COPYBOOK GENERATED

```
 01 SCR306A.
    05 SCR306A-DATE-ATTR-STD       PIC X(1).
    05 SCR306A-DATE-ATTR-EXT       PIC X(1).
    05 SCR306A-DATE-ATTR-COL       PIC X(1).
    05 SCR306A-DATE-ATTR-CUR       PIC X(1).
    05 SCR306A-GROUP-001           OCCURS 15 TIMES
         10 SCR306A-FNC-ATTR-STD  PIC X(1).
         10 SCR306A-FNC-ATTR-EXT  PIC X(1).
         10 SCR306A-FNC-ATTR-COL  PIC X(1).
         10 SCR306A-FNC-ATTR-CUR  PIC X(1).
         10 SCR306A-FNC-DATA      PIC X(1).
         10 SCR306A-MAKE-ATTR-STD PIC X(1).
         10 SCR306A-MAKE-ATTR-EXT PIC X(1).
         10 SCR306A-MAKE-ATTR-COL PIC X(1).
         10 SCR306A-MAKE-ATTR-CUR PIC X(1).
         10 SCR306A-MAKE-DATA     PIC X(20).
      05 SCR306A-MSG-ATTR-STD     PIC X(1).
      05 SCR306A-MSG-ATTR-EXT     PIC X(1).
```

```
05 SCR306A-MSG-ATTR-COL      PIC X(1).
05 SCR306A-MSG-ATTR-CUR      PIC X(1).
05 SCR306A-MSG-DATA          PIC X(72).
```

JCL Samples - JCL401M3

```
//    JOB
//* ---------------------------------------------------------------- *
//*       TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ---------------------------------------------------------------- *
//*
//* JCL: JCL401M3
//*
//* DESCRIPTION: GENERATE COBOL OR PL1 COPYCODE MEMBERS TO REFLECT
//*              THE FIELD LAYOUT ON A PARTICULAR SCREEN.
//*
//* NOTES:
//*
//*
//* ---------------------------------------------------------------- *
//*      **********************************************************
//*      ** GENERATE COPYCODE                                    **
//*      **    - COPYLIB DD IS USED TO CHECK IF MEMBER EXISTS  **
//*      **    - SET FUNCTION TO THE FOLLOWING:                **
//*      **        FUNC = SCR2CBL  FOR COBOL COPY BOOK         **
//*      **        FUNC = SCR2PLI  FOR PLI COPY BOOK           **
//*      **    - CONTROL CARD                                  **
//*      **         COL 01-08 = SCREEN NAME                    **
//*      **********************************************************
//STA    EXEC TBLDB2,
//         FUNC=SCR2CBL                  ==> TAS SCREEN COPY BOOK
//COPYLIB  DD  DSNAME=SSI.TEST.COPYLIB,DISP=SHR
//COPYOUT  DD  DSN=&&COPYDB,DCB=BLKSIZE=80,
//         SPACE=(TRK,(5,1)),UNIT=SYSDA,DISP=(NEW,PASS)
//CONTROL  DD  *
SCR306A
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
//*      **********************************************************
//*      ** MOVE COPYCODE TO LIBRARY                             **
//*      **    - SYSUT1 AND SYSUT2 SHOULD BE YOUR OUTPUT       **
//*      **      LIBRARY AND THE SAME AS COPYLIB IN STEP A.    **
//*      **********************************************************
//STB    EXEC PGM=IEBUPDTE,PARM=MOD,REGION=512K,COND=(5,LE)
//SYSPRINT  DD  SYSOUT=*
//SYSUT1    DD  DSNAME=SSI.TEST.COPYLIB,DISP=SHR
//SYSUT2    DD  DSNAME=SSI.TEST.COPYLIB,DISP=SHR
//SYSIN     DD  DSN=&&COPYDB,DISP=(OLD,DELETE)
```

JCL401M3 - Error Messages Generated

CONDITION CODE 00

- 'SCREEN RECORD NOT FOUND FOR SCREEN ---> xxxxxxxx'
  'PROCESSING TERMINATES
  A screen name was specified and no screen exists in TABLES/AS for that name. Also, the plan could be pointing to a different set of SSI control tables. Determine cause, fix & resubmit.
- 'FIELD NAMES NOT FOUND FOR SCREEN ---> xxxxxxxx'
  'PROGRAM WILL GENERATE NAMES'
  This is an informational message only. If your screen does have screen names defined then it is an error and you should notify SSI.
- 'ERROR READING SCREEN DATA BASE'
  'IMS RETURN CODE ---> xx'
  Contact SSI. There has been an unexpected error encountered trying to read the Screen or Names rows for the SCREENDB.
  'OPEN OF COPYLIB DD FAILED' - Insure names in JCL are correct and then rerun.
- 'DIRECTORY ERROR READING PDS' - Contact SSI. Unexpected error encountered accessing the Copylib directory.
- 'INSUFFICIENT REGION SIZE'
  'INCREASE STEP REGION SIZE' - Fix and rerun.
  'UNDETERMINED COPYLIB ERROR' - Contact SSI. Unexpected error encountered accessing Copylib.
- 'CLOSE OF COPYLIB DD FAILED' - Contact SSI.
- '***ERROR*** NOT ENOUGH FIELDS TO FINISH ARRAY FOR SCREEN xxxxxx'
- FIRST MISSING FIELD NAME = xxxxxxxx'
  'LAST FIELD NAME PROCESSED = xxxxxxxx'
  Contact SSI.
- '***ERROR*** INVALID ARRAY PATTERN FOR SCREEN xxxxxxxx'
  'FIELD NAME FOUND = xxxxxxxx'
  'SHOULD BE = xxxxxxxx'
  Contact SSI.

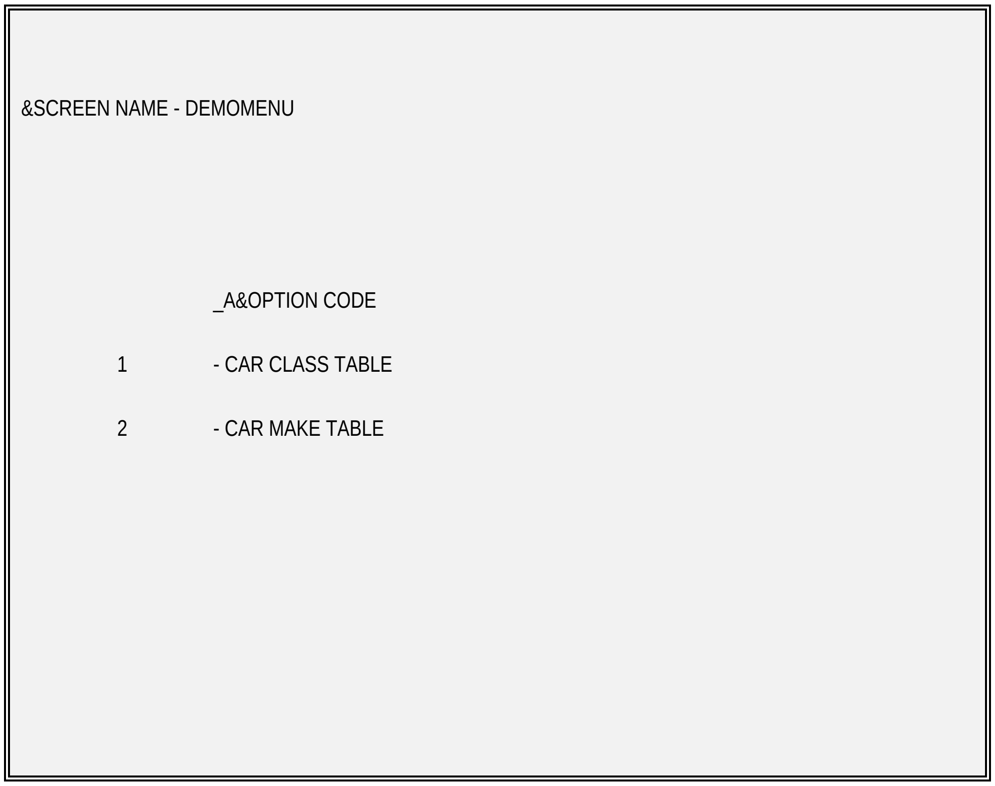
JCL401M4 - Screen Image Print

This utility uses DB2 plan TBLUTIL.

It produces a hardcopy printout of the image of a TABLES/AS screen. The image printed is the same as if the screen were retrieved in change mode and displayed with the Format option. Multiple screens may be printed by supplying multiple control cards.

SAMPLE SCREEN IMAGE PRINT

CONTROL CARD #001 - DEMOMENU



```
&SCREEN NAME - DEMOMENU




                         _A&OPTION CODE

             1           - CAR CLASS TABLE

             2           - CAR MAKE TABLE
```

JCL Samples - JCL401M4

```
//    JOB
//* ---------------------------------------------------------------- *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ---------------------------------------------------------------- *
//*
//* JCL: JCL401M4
//*
//* DESCRIPTION: TABLES/AS SCREEN IMAGE PRINT.
//*
//*
//* NOTES:
//*
//*
//* ---------------------------------------------------------------- *
//*     ********************************************************
//*     ** PRINT SCREENS                                      **
//*     **   - CONTROL CARD                                   **
//*     **       COL 01    = 'C'                              **
//*     **       COL 02-09 = SCREEN NAME                      **
//*     ********************************************************
//STA    EXEC TBLDB2,
//         FUNC=SCRNPRNT                 ==> SCREEN IMAGE PRINT
//EXTRACT  DD  DUMMY
//CONTROL  DD  *
CDEMOMENU
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
```

JCL401M4 - Error Messages Generated

CONDITION CODE 00

- 'E001--NAME OF SCREEN TO BE EXTRACTED MISSING' - Screen name in control record, pos 2-9, is blanks
- 'E002--INVALID EXTRACT FUNCTION CODE' - Col 1 of control record does not contain a 'C'.

CONDITION CODE 08

- 'E004--SCREEN TO BE EXTRACTED NOT FOUND' - The screen name in control record, pos 2-9, does not have a corresponding screen in the TABLES/AS SCREENDB. Check your screen name

and/or check that the TBLUTIL plan is bound to the correct
set of SSI control tables.

JCL401M7 - Create Screen Load Module

Create screen load module uses DB2 plan TBLUTIL.

This job produces hardcopy documentation regarding a screen while creating a load module for the screen. The screen load module name must be the same as the screen name. There are a number of ways to access a TABLES/AS defined screen. If a load module exists, in all cases other than using Option 9 - Screen Processor, the load module will be used for the screen and the remaining SSI control tables will be accessed from memory. When using Option 9, screen load modules are not used and all SSI control tables are read directly from DB2, not from memory resident preloaded tables.

An assembler module is created and then link edited. The load module created includes the information defined in TABLES/AS using the Screen Design functions, the Edit Definition functions and the Mapping functions. All other **TABLES/AS** defined information will be retrieved from preloaded SSI control tables whenever the load module is accessed.

The screen load module name is not needed in the Screen Processor DB2 plan or its equivalent.

The printed documentation includes:

- A screen image print.
- A list of each field on the screen showing the field name, the location on the screen, screen field type, screen field length, edits defined for the field and relational edits if the field is specified as a relational control field.
- A list of each map in the screen defining the table used to create the map, the map name, the source of the table definition, editor module name if used, the static SQL module name if used, the uses of the map (process options, cross table validation, etc.) with the defined conditions and the correlation of screen and field names used in the map.

CICS only____

The screen load module must be defined to CICS (e.g., PPT or RDO).
____**End CICS only**

JCL Samples - JCL401M7 for IMS

```
//    JOB
//* ---------------------------------------------------------------- *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ---------------------------------------------------------------- *
//*
//* JCL: JCL401M7
//*
//* DESCRIPTION: CREATE SCREEN LOAD MODULE WITH DOCUMENTATION
//*
//*
//* NOTES: - THE SCREEN NAME MUST BE CHANGED IN STEP STA, STEP STD
//*          AND IN THE SYSLMOD DD CARD IN THE LAST STEP.
//*        - CHECK THE IMS LIBRARY IN THE LAST STEP.
//*
//*
//* ---------------------------------------------------------------- *
//*     ********************************************************
//*     ** CREATE COMPOSITE RECORD                            **
//*     **    - EXTRACT AND EXTRACT2 ARE THE OUTPUT FILES.    **
//*     **    - CONTROL CARD                                  **
//*     **        COL 01-08 = SCREEN NAME                     **
//*     ********************************************************
//STA    EXEC TBLDB2,
//        FUNC=COMPOSIT              ==> CREATE COMPOSITS
//EXTRACT  DD  DSN=&&EXTRACT,DISP=(NEW,PASS),
//        DCB=(RECFM=FB,LRECL=157,BLKSIZE=4710),
//        UNIT=SYSDA,SPACE=(TRK,(5,1))
//EXTRACT2 DD  DSN=&&EXTRACT2,DISP=(NEW,PASS),
//        DCB=(RECFM=FB,LRECL=157,BLKSIZE=4710),
//        UNIT=SYSDA,SPACE=(TRK,(5,1))
//CONTROL  DD  *
SCRN001
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
//*
//*     ********************************************************
//*     ** PRINT THE COMPOSITE RECORDS                        **
//*     ********************************************************
//STB    EXEC TBLBATCH,
//        FUNC=PRTEDITS
//EXTRACT  DD DSN=&&EXTRACT,DISP=(OLD,DELETE)
//*     ********************************************************
//*     ** PRINT SCREEN MAPPING DOCUMENTATION                 **
//*     ********************************************************
```

```
//STC    EXEC TBLDB2,
//        FUNC=PRTMAPS
//EXTRACT2 DD DSN=&&EXTRACT2,DISP=(OLD,DELETE)
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
//*
//*      ********************************************************
//*      ** EXTRACT COMPOSITE RECORDS                          **
//*      **    - CONTROL CARD                                  **
//*      **         COL 01-08 = SCREEN NAME                    **
//*      ********************************************************
//STD    EXEC TBLDB2,COND=(5,LE),
//        FUNC=COMEXTCT                  ==> EXTRACT COMPOSITS
//EXTRACT  DD  DSN=&&EXTRACT,DISP=(NEW,PASS),
//        DCB=(BLKSIZE=8000,RECFM=FB,LRECL=8000),
//        UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//CONTROL  DD  *
SCRN001
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
//*      ********************************************************
//*      ** BUILD COMPOSITE SOURCE                             **
//*      ********************************************************
//STE    EXEC TBLBATCH,
//        FUNC=COMSOURC
//INFILE    DD  DSN=&&EXTRACT,DISP=(OLD,DELETE)
//OUTFILE   DD  DSN=&&OUTFILE,DISP=(NEW,PASS),
//         DCB=(RECFM=FB,BLKSIZE=3120,LRECL=80),
//         UNIT=SYSDA,SPACE=(TRK,(1,1),RLSE)
//*
//*      ********************************************************
//*      ** ASSEMBLE AND LINK THE SOURCE                       **
//*      **    - SYSLMOD MUST BE CHANGED TO REFLECT THE        **
//*      **      CORRECT SCREEN NAME                           **
//*      **    - REVIEW C.SYSLIB AND L.SYSLIB DD CARDS         **
//*      ********************************************************
//STF      EXEC  ASMHCL,PARM.L='LIST,LET,REUS,AMODE=31,RMODE=ANY',
//        COND=(5,LE)
//C.SYSLIB   DD  DSN=SYS1.MACLIB,DCB=BLKSIZE=8000
//          DD  DSN=SSI.TEST.SOURCE,DISP=SHR
//C.SYSIN    DD  DSN=&&OUTFILE,DISP=(OLD,DELETE)
//L.SYSLMOD  DD  DSN=SSI.TEST.LOADLIB(SCRN001),DISP=SHR,UNIT=,VOL=
//L.SYSLIB   DD  DSN=SSI.TEST.LOADLIB,DISP=SHR
//          DD  DSN=IMSVS.RESLIB,DISP=SHR
```

JCL : CREATE SCREEN LOAD MODULE (IMS)

JCL Samples - JCL401M7 for CICS

```
//    JOB
//* ------------------------------------------------------------------ *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE INTERNATIONAL
//* ------------------------------------------------------------------ *
//*
//* JCL: JCL401M7
//*
//* DESCRIPTION: CREATE SCREEN PROGRAM WITH DOCUMENTATION
//*
//*
//* NOTES: - THE SCREEN NAME MUST BE CHANGED IN STEP STA, STEP STD
//*          AND IN THE SYSLMOD DD CARD IN THE LAST STEP.
//*        - IN THE LAST STEP (ASM AND LINK), THE CICS LIBRARIES
//*          MUST BE CHECKED.
//*
//* ------------------------------------------------------------------ *
//*      *******************************************************
//*      ** CREATE COMPOSITE RECORD                           **
//*      **    - EXTRACT AND EXTRACT2 ARE THE OUTPUT FILES.    **
//*      **    - CONTROL CARD                                 **
//*      **         COL 01-08 = SCREEN NAME                   **
//*      *******************************************************
//STA    EXEC TBLDB2,
//         FUNC=COMPOSIT                  ==> CREATE COMPOSITS
//EXTRACT  DD  DSN=&&EXTRACT,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(TRK,(5,1)),
//         DCB=(RECFM=FB,LRECL=157,BLKSIZE=4710)
//EXTRACT2 DD  DSN=&&EXTRACT2,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(TRK,(5,1)),
//         DCB=(RECFM=FB,LRECL=157,BLKSIZE=4710)
//CONTROL  DD  *
DB2TE001
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2X)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
//*
//*******************************************************
//*    PRINT THE COMPOSITE RECORDS
//*******************************************************
//STB    EXEC TBLBATCH,
//         FUNC=PRTEDITS
//EXTRACT  DD DSN=&EXTRACT,DISP=(OLD,DELETE)
//*
//*******************************************************
//*    PRINT SCREEN MAPPING DOCUMENTATION
```

```
//*********************************************************
//STC    EXEC TBLDB2,
//         FUNC=PRTMAPS
//EXTRACT2 DD DSN=&&EXTRACT2,DISP=(OLD,DELETE)
//SYSTSIN  DD  *
  DSN SYSTEM(DB2X)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
//*
//*      *********************************************************
//*      ** EXTRACT COMPOSITE RECORDS                         **
//*      **    - CONTROL CARD                                 **
//*      **         COL 01-08 = SCREEN NAME                   **
//*      *********************************************************
//STD    EXEC TBLDB2,COND=(5,LE),
//         FUNC=COMEXTCT                  ==> EXTRACT COMPOSITS
//EXTRACT  DD  DSN=&&EXTRACT3,DISP=(NEW,PASS),
//         DCB=(BLKSIZE=8000,RECFM=FB,LRECL=8000),
//         UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//CONTROL  DD  *
DB2TE001
/*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2)
  RUN PROGRAM(TBLUTIL) PLAN(TBLUTIL)
  END
/*
//*      *********************************************************
//*      ** BUILD SOURCE PROGRAM FOR THE SCREEN               **
//*      *********************************************************
//STE    EXEC TBLBATCH,
//         DBHLQ='SSI.TEST',              ==> DB HI-LEVEL QUALIFIER
//         FUNC=COMSOURV                  ==> BUILD COMPOSIT SOURCE
//INFILE    DD  DSN=&&EXTRACT3,DISP=(OLD,DELETE)
//OUTFILE   DD  DSN=&&OUTFILE,DISP=(NEW,PASS),
//          DCB=(RECFM=FB,BLKSIZE=3120,LRECL=80),
//          UNIT=SYSDA,SPACE=(TRK,(1,1),RLSE)
//*
//*      *********************************************************
//*      ** ASSEMBLE AND LINK THE SOURCE                      **
//*      **    - SYSLMOD MUST BE CHANGED TO REFLECT THE       **
//*      **      CORRECT SCREEN NAME                          **
//*      **    - REVIEW C.SYSLIB AND L.SYSLIB DD CARDS        **
//*      **    - CHECK CICS LIBRARIES FOR CORRECT DSNAMES     **
//*      *********************************************************
//STF     EXEC  ASMHCL,PARM.L='LIST,LET,REUS,AMODE=31,RMODE=ANY',
//        COND=(5,LE)
//C.SYSLIB  DD  DSN=SYS1.MACLIB,DISP=SHR,DCB=BLKSIZE=8000
//          DD  DSN=SSI.TEST.SOURCE,DISP=SHR
//          DD  DSN=CICS.MACLIB,DISP=SHR
```

```
//C.SYSIN   DD  DSN=&&OUTFILE,DISP=(OLD,DELETE)
//L.SYSLIN  DD  DSN=CICS.MACLIB(DFHEILIA),DISP=SHR
//          DD  DSN=&&OBJ,DISP=(OLD,DELETE)
//L.SYSLMOD DD  DSN=SSI.TEST.LOADLIB(DB2TE001),DISP=SHR,UNIT=,VOL=
//L.SYSLIB  DD  DSN=SSI.TEST.LOADLIB,DISP=SHR
//          DD  DSN=CICS.LOADLIB,DISP=SHR
```

JCL : CREATE SCREEN LOAD MODULE (CICS)

JCL401M7 - Error Messages Generated

COMPOSITE STEP

CONDITION CODE 12 - See message printed at end of report. Contact SSI if this condition code occurs.

PRTEDITS STEP

CONDITION CODE 00

'INVALID RECORD TYPE 'extract record' - If this message is displayed, contact SSI. An extract record does not have an ID of 00-13.

PRTMAPS STEP

CONDITION CODE 00

'INVALID RECORD TYPE 'extract record' - If this message is displayed, contact SSI. An extract record does not have an ID of 00 or 01.

COMEXTCT STEP

CONDITION CODE 08 - Screen named in control record not found. Make sure the screen name entered is correct. Make sure the screen exists in TABLES/AS. Make sure the plan is pointing to the correct set of SSI control tables.

COMSOURC/COMSOURV STEP

CONDITION CODE 04

'SS8500M6 ERROR - INVALID OR NO PARM SPECIFIED, ENDING WITH RC=4

CONDITION CODE 08

'SS8502M6 ERROR - THERE IS NO INPUT DATA TO PROCESS, ENDING WITH RC=8 INFILE was empty. Check prior steps and/or dataset name.

CONDITION CODE 12

'SS8500M6 ERROR - COULD NOT OPEN THE INPUT FILE, ENDING WITH RC=12. Check INFILE dataset name.

CONDITION CODE 16

'SS8500M6 ERROR - COULD NOT OPEN THE OUTPUT FILE, ENDING WITH RC=16. Check OUTFILE dataset name and specification.

CONDITION CODE 99

'SS8500M6 ERROR - INTERNAL ERROR, CHECK R TYPE PROCESSING.
ENDING WITH RC=99

Extract program is passing an invalid record type. Contact SSI.

Section 5

TABLES/AS Technical Guide

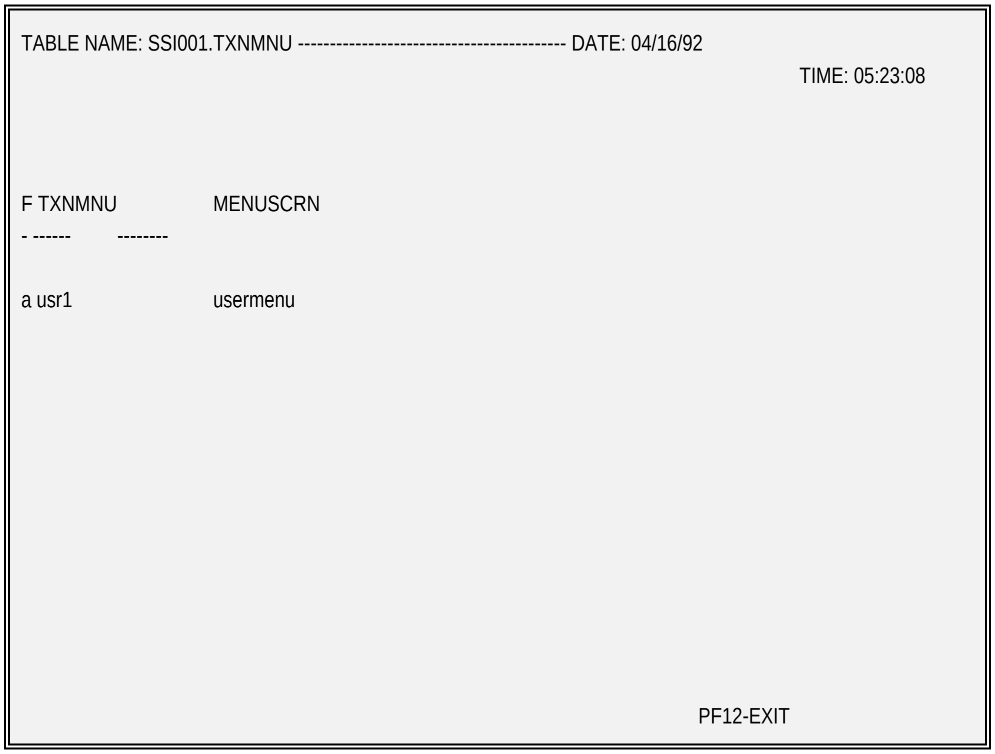
Custom Screen Processor Transaction Code

Custom Transaction Code

Through the use of the Transaction Menu Control Table (TXNMNU Control table), it is possible to define a transaction code that allows you to switch to a User menu screen. You would use this facility if you wished to process an application without starting at the **TABLES/AS** Screen Processor Menu (TS09). See Appendix L for additional details.

For example, if you wish to start a menu screen called USERMENU using a transaction code USR1, the following steps must be taken:

1. Update TXNMNU Control table using **TABLES/AP** as shown below:



```
TABLE NAME: SSI001.TXNMNU ----------------------------------------- DATE: 04/16/92
                                                                                  TIME: 05:23:08



F TXNMNU              MENUSCRN
- ------        --------

a usr1                usermenu













                                                                        PF12-EXIT
```

2. Update the IMS GEN or CICS PCT with the following information:

IMS:		CICS:	
TXN Code	= USR1	TXN Code	= USR1
PSB Name	= TSIM0900	Program Name	= TSCM0900
Program Name	= TSIM0900		

3. You must provide a plan name (may be TSCM0900) for the given transaction in the Resource Control Table for CICS.

4. If DB2 plan TSIM0900/TSCM0900 is to be used, any new DBRMs used by the TXN must be added to the plan and the plan rebound.

5. If a unique DB2 plan will be created for this TXN, in addition to DBRMs created for this TXN, all of the DBRMs in TSIM0900/TSCM0900 must be included in this new plan.

Section 6

TABLES/AS Technical Guide

Screen Programming

Interface

Introduction

The **TABLES/AS** Screen Processor meets most on-line application needs without requiring any programming. However, to provide maximum flexibility, the **TABLES/AS** Screen Generator, data editing, and data mapping facilities can also be accessed through editor programs or through application programs. Editor programs are discussed in the section entitled Online Maintenance.

Facilities are provided for application programs to utilize **TABLES/AS** defined screens to perform;

- Simple screen input and output
- Screen input and output with data edited
- Screen I/O with data edited and mapped where "Mapped" is the process of moving data between screen and record areas performing various data conversions.

The following pages define the general requirements and functions provided followed by IMS and then CICS specifics. All examples shown are written in COBOL.

General Requirements and Functions

Steps Required

Step 1 is to design a screen using the **TABLES/AS** development functions. Depending on the amount of work you want performed by **TABLES/AS** you would complete;

- Screen Design - for simple screen input and output
- Edit Definitions - to perform field and relational edits
- Mapping - to perform data movement and formatting.

Step 2 is to create the copy book for the screen using the batch facility provided with **TABLES/AS** (JCL401M3).

A simple screen with only two fields, employee name and social security number, would generate the following COBOL copy code:

```
01  EMPLOYEE.
    05   EMPLOYEE-SSNO-ATTR-STD       PIC X(1).
    05   EMPLOYEE-SSNO-ATTR-EXT       PIC X(1).
    05   EMPLOYEE-SSNO-ATTR-COL       PIC X(1).
    05   EMPLOYEE-SSNO-ATTR-CUR       PIC X(1).
    05   EMPLOYEE-SSNO-DATA                PIC X(9).
    05   EMPLOYEE-NAME-ATTR-STD       PIC X(1).
    05   EMPLOYEE-NAME-ATTR-EXT       PIC X(1).
    05   EMPLOYEE-NAME-ATTR-COL       PIC X(1).
    05   EMPLOYEE-NAME-ATTR-CUR       PIC X(1).
    05   EMPLOYEE-NAME-DATA                PIC X(25).
```

There will be five fields in the copy code for each field on the screen. One field will contain the data itself. The other four fields are used for screen attribute characters.

STD (Standard Field Attribute)
Used to change the standard attribute of a field dynamically, such as protecting a field that was previously unprotected.

EXT (Extended Field Attribute)
Used to change the extended attribute of a field. For example, to set a field to blink which previously did not blink. This option works only if the user terminal supports extended field attributes.

COL (Color Field Attribute)
Used to change the color of a field. For example, to change a field that displayed in green to display in yellow. This option works only if the terminal supports color attributes.

CUR (Cursor Field Attribute)
Used to indicate that the cursor should be placed in the data field when the screen is displayed. A value of E is used for the cursor attribute; any non-blank character may be used.

A list of all the values for the attribute characters and meanings can be found in the Appendix B.

<u>Step 3</u> is to create the screen load module using the batch facility provided with **TABLES/AS** (JCL401M7). The screen program does all of the screen processing defined in **TABLES/AS** including the physical screen I/O, the data editing and the data mapping. Every screen has its own corresponding screen program with the same name. Any user-written program which requires screen processing simply links to this screen program.

The load module form of this program can be stored in any PDS library defined in the linkage editor JCL. Normally this program is used as a sub-routine of the user-written main line program. Therefore, it should be stored in an on-line library.

<u>Step 4</u> is to write the application program utilizing the screen programming interface.

Simple Screen I/O

After a **TABLES/AS** screen is designed and a load module created, an application program can read and write that screen. Before writing the screen, the screen copy code area could be populated with data by this application program. After reading it, the program could perform editing and any other formatting and moves to/from the record I/O area.

Reading and writing are performed in IMS with functions GU, GN and ISRT. In CICS, the functions are GET and PUT.

Data Editing

After designing a screen, the next step is to tell **TABLES/AS** how each field should be edited. The editing can be simple to complex, from making sure a value is equal to Y or N, to relational editing, where many fields on the screen participate in determining the validity of a field.

Once the editing rules are defined, they are available to the **TABLES/AS** interface. A screen is edited by calling the screen program with a function of GNE or GUE in IMS and GETE in CICS. These functions cause the **TABLES/AS** interface to read the screen and map each field into the screen copy code area and then check that each field passes the edits that were defined. If a field is found to be in error, it is highlighted and the error-flag switch is set to Y. If an error message is defined, it will be moved to the error message field.

In IMS, the following procedure is most often used to handle data editing. A screen is sent using the ISRT function. The screen is read back in and edited using the GNE or GUE function. If the EDIT-ERROR-FLAG is equal to N, then there is no error in the data and processing can continue. If the EDIT-ERROR-FLAG is equal to Y, then there is an error. The next step is to send the screen back out using the ISRT function. This process should continue until the EDIT-ERROR-FLAG is returned with a value of N or the user presses a programmed PF key.

In CICS, the following procedure is most often used to handle data editing. A screen is sent using the PUT function. The screen is read back in and edited using the GETE function. If the EDIT-ERROR-FLAG is equal to N, then there is no error in the data and processing can continue. If the EDIT-ERROR-FLAG is equal to Y, then there is an error. The next step is to send the screen back out using the PUT function. This process should continue until the EDIT-ERROR-FLAG is returned with a value of N or the user presses a programmed PF key.

GNE, GUE, GETE OPERATIONS

These operations cause the screen to be read and the data on the screen to be mapped into the screen copy code area. Fields will be filled with blanks when nothing is entered in them. The return code should be checked for a value of zero to insure that the operation was successfully completed. Each field is edited according to the defined edit rules. If an error is found, the edit error flag is set to Y and the field is highlighted. The entire screen is edited during this operation.

Data Mapping

Data mapping is the **TABLE/AS** process of moving data from a screen copy code field to its corresponding field in a table input or output area. This process includes the conversion of data from its external display format to its internal table storage format. Field conversions supported include:

- Conversion of date formats
- Conversion of alphanumeric formats
- Conversion of decimal point
- Conversion of negative sign
- Conversion of storage formats (packed, binary, zoned, character)

TABLES/AS handles fields that occur more than one time on a screen or in a table. For example, if a particular field occurred five times on a screen, the **TABLES/AS** interface can be instructed to map each occurrence separately, or to map all five occurrences at once.

TABLES/AS will map information from the screen copy code area to a table input or output area by use of the MAPI function code. Mapping from the table input or output area to the screen copy code area is achieved with the MAPO function code.

All mapping rules are defined on-line after the screen has been designed. When the **TABLES/AS** screen program is assembled, it will include these mapping rules, eliminating them from your program.

Prior to use of MAPI, the screen must be read with Data Editing.

SAMPLE FIELD CONVERSIONS

SCREEN FIELD	TABLE FIELD

1. Alphanumeric (A) character field without special format converted to a character field (C). (No conversion required.)

ABCDEF	ABCDEF

2. Alphanumeric (A) character field with a special format converted to a character field (C).

243-22-1661	243221661

3. Alphanumeric (A) character field with a special format converted to a numeric field, either zoned decimal (D), packed decimal (P), or binary (B).

243-22-1661	243221661

4. Quantity (Q) field converted to another numeric format such as zoned decimal (D), packed decimal (P), or binary (B). In addition, the sign is normalized and the decimal point is aligned. Assume pic clause was S999V99.

-15.25	0152N

5. Numeric (N) field on screen converted to another format in the table such as character (C), zoned decimal (D), packed decimal (P), or binary (B).

243221661	243221661

6. Date (D) field on screen converted to another date format and also another storage format such as character (C), zoned decimal (D), packed decimal (P), or binary (B).

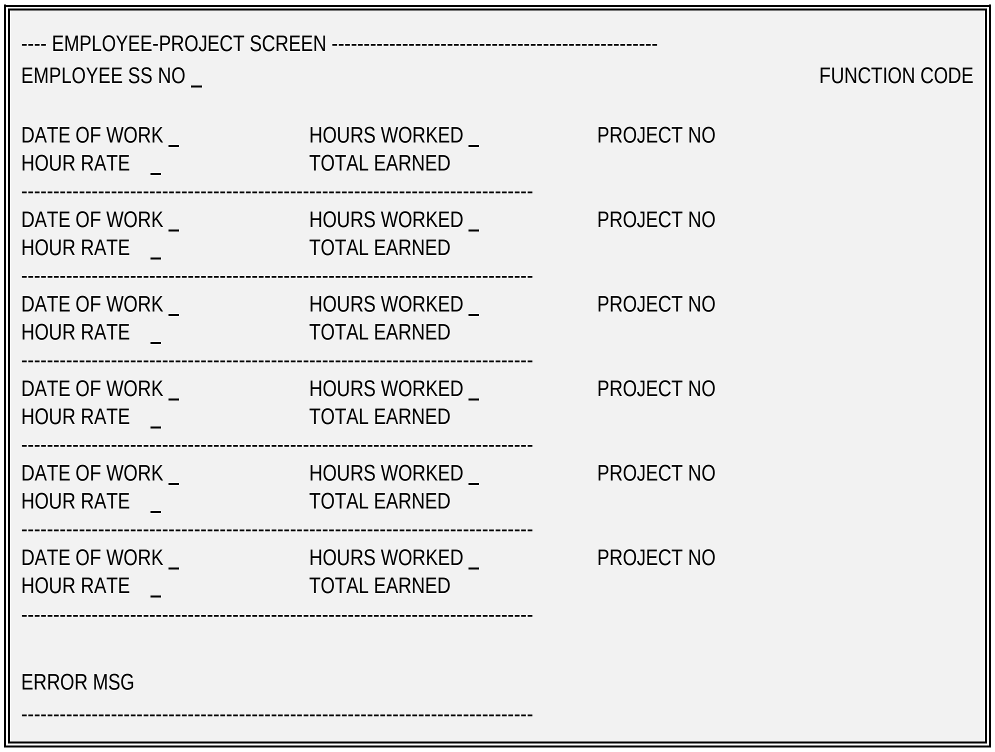
12/31/85	85365

FIELDS WHICH OCCUR MULTIPLE TIMES

A field is often repeated several times on a screen. There are two ways that these fields could be mapped. The method depends on how the records are structured in the database. Each field could be mapped into a separate record in a database, or all occurrences of the field could be mapped into one record in the database.

To map all of the occurrences of a screen field into one record, all of the occurrences of the field must be selected for mapping. Otherwise, **TABLES/AS** will map each occurrence separately.

In order to map in a specific occurrence of a screen row, the MAP-OCC-NO field in the CONTROL-AREA should be set to the relative number of the occurrence. For example, to map in the second row on the screen, the Number 2 should be moved to the MAP-OCC-NO field. When the **TABLES/AS** interface is called, the second row will be mapped in.



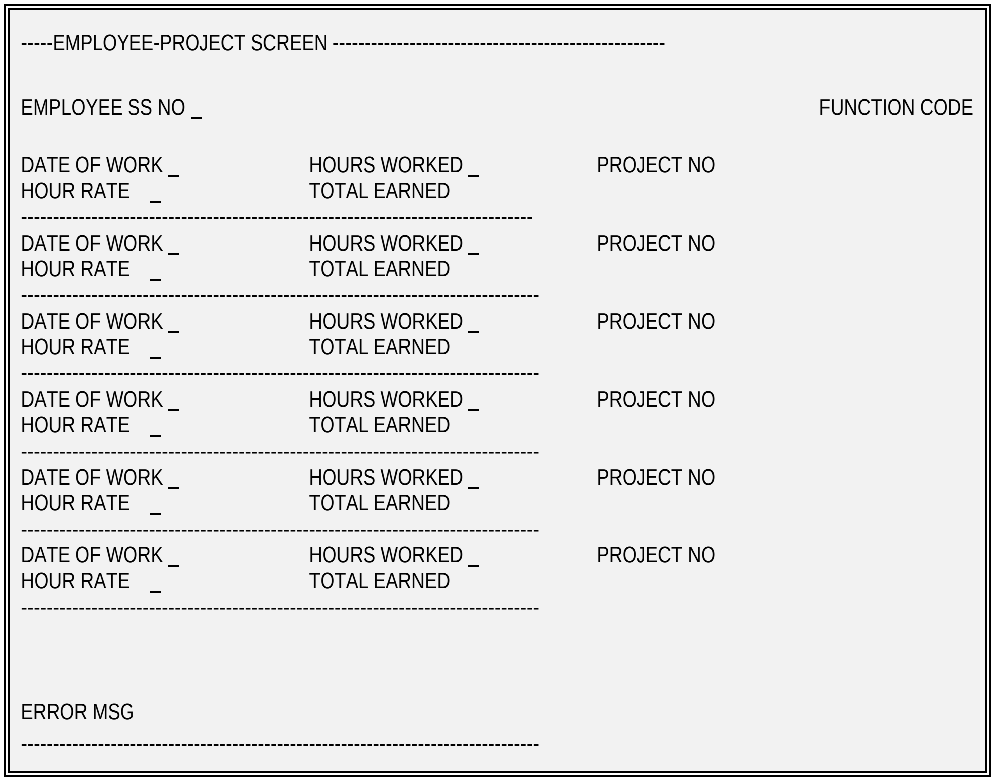
```
---- EMPLOYEE-PROJECT SCREEN --------------------------------------------------
EMPLOYEE SS NO _                                                    FUNCTION CODE

DATE OF WORK _          HOURS WORKED _          PROJECT NO
HOUR RATE    _          TOTAL EARNED
-------------------------------------------------------------------------------
DATE OF WORK _          HOURS WORKED _          PROJECT NO
HOUR RATE    _          TOTAL EARNED
-------------------------------------------------------------------------------
DATE OF WORK _          HOURS WORKED _          PROJECT NO
HOUR RATE    _          TOTAL EARNED
-------------------------------------------------------------------------------
DATE OF WORK _          HOURS WORKED _          PROJECT NO
HOUR RATE    _          TOTAL EARNED
-------------------------------------------------------------------------------
DATE OF WORK _          HOURS WORKED _          PROJECT NO
HOUR RATE    _          TOTAL EARNED
-------------------------------------------------------------------------------
DATE OF WORK _          HOURS WORKED _          PROJECT NO
HOUR RATE    _          TOTAL EARNED
-------------------------------------------------------------------------------


ERROR MSG
-------------------------------------------------------------------------------
```

DATA INDEPENDENCE

TABLES/AS facilitates data independence when mapping records with multiple occurrences. A program can be coded without any knowledge of how many fields exist on the screen. For example, the screen format displayed below has six occurrences of the EMPLOYEE-PROJECT group. A program may be written as if the number of occurrences is unknown. The **TABLES/AS** interface will return data in the record input or output area until the seventh call is made. Since there is no seventh occurrence, the screen program will return a condition code of 26 to indicate the out of bounds condition.

The screen itself is another area of data independence. The format of the screen can be changed without changing the screen copy code. The reverse is also true. The fields in the screen copy code area can be re-sequenced without re-sequencing the fields on the screen. Be careful to use the E cursor positioning method in this case or the cursor may not be positioned where desired.

NOTE: If the screen copy code has been resequenced in **TABLES/AS**, whenever the screen is saved with PF3/PF4 while in change a screen format, you will need to resequence the copy code again. Each time a screen format is saved with PF3/PF4, the resultant copy code will be in screen field occurrence sequence.


```
-----EMPLOYEE-PROJECT SCREEN ----------------------------------------------------

EMPLOYEE SS NO _                                                    FUNCTION CODE

DATE OF WORK _            HOURS WORKED _            PROJECT NO
HOUR RATE    _            TOTAL EARNED
--------------------------------------------------------------------------------
DATE OF WORK _            HOURS WORKED _            PROJECT NO
HOUR RATE    _            TOTAL EARNED
--------------------------------------------------------------------------------
DATE OF WORK _            HOURS WORKED _            PROJECT NO
HOUR RATE    _            TOTAL EARNED
--------------------------------------------------------------------------------
DATE OF WORK _            HOURS WORKED _            PROJECT NO
HOUR RATE    _            TOTAL EARNED
--------------------------------------------------------------------------------
DATE OF WORK _            HOURS WORKED _            PROJECT NO
HOUR RATE    _            TOTAL EARNED
--------------------------------------------------------------------------------
DATE OF WORK _            HOURS WORKED _            PROJECT NO
HOUR RATE    _            TOTAL EARNED
--------------------------------------------------------------------------------



ERROR MSG
--------------------------------------------------------------------------------
```

Conversational / Non-Conversational IMS Transactions

CONVERSATIONAL TRANSACTIONS

Conversational processing allows the person at the terminal to have several interactions with the application program. When the person enters a transaction code that has been defined as conversational, IMS schedules the conversational program that corresponds to that transaction code. The person has to enter the transaction code only on the first screen. The data in the subsequent screens are treated as a continuation of the conversation started on the first screen.

The program must be able to retain certain information between requests in order for the program to continue the conversation. IMS stores that information in the scratch pad area (SPA). The conversation will continue until blanks are moved to the first eight bytes of the SPA, and the SPA is returned to IMS.

NON-CONVERSATIONAL TRANSACTIONS

No SPA is used in this type of program. Any data that is needed between requests must be stored elsewhere such as in some darkened fields on the screen. The Set Transaction command must be entered before starting the conversation. This allows the conversation to continue without requiring the transaction code to be continually entered.

Once the Set Transaction command has been entered, the screen should be cleared of any data and re-entered as a blank screen. This second screen will be received by the program as all blanks. The processing of screens from this point on may proceed as if the non-conversational program was a conversational program. A null screen should be sent to the terminal to end the conversation. This will set the terminal to blanks and allow the IMS command, Reset, to be entered so the terminal can be used for other transactions.

IMS CALL FORMATS

For Screen I/O and Data Editing
```
CALL 'SCRNPROG' USING   CONTROL-AREA
                        SCREEN-COPY-CODE-AREA
                        LOGICAL-TERMINAL-PCB.
```

For Data Mapping after Data Editing
```
CALL 'SCRNPROG' USING   CONTROL-AREA
                        SCREEN-COPY-CODE-AREA
                        RECORD-INPUT-OUTPUT-AREA.
```

CONTROL-AREA
- Contains all control information passed to and from the screen program.
- Layout of the CONTROL-AREA in COBOL record structure is as follows:

```
01   CONTROL-AREA.
 05  SCREEN-FUNCTION                     PIC X(4).
 05  PFK-PAK-AREA                        PIC 9(2).
 05  CURSOR-POSITIONING                  PIC 9(4) COMP.
 05  RETURN-CC                           PIC 9(2).
 05  EDIT-ERROR-FLAG                     PIC X(1).
 05  SEGMENT-TABLE-NAME                  PIC X(8).
 05  DBD-NAME                            PIC X(8).
 05  MAP-OCC-NO                          PIC 9(4) COMP.
```

SCREEN-FUNCTION

- ISRT = Insert the message using the screen copy code area for output data. The previous call to IMS must have been an ISRT to SPA for conversational.
  Causes the screen to be written.

- GNE = (Get Next message and edit the data) used by conversational transactions to read an IMS message and map the data into the screen copy code area. Each field is edited. The previous call to IMS must have been a GU to the SPA.

- GUE = (Get Unique message and edit the data) is used by non-conversational transactions to read an IMS message and map the data into the screen copy code area. Each field is edited.

• MAPI =	Map a given occurrence of a file or a table from the screen copy code area to the record input/output area. The previous contents of other fields in the input/output area are not destroyed. It is possible to map into a record area which already contains some data and the MAPI function is called to replace the value in some fields, but not all.
• MAPO =	Map a given occurrence of a file or of a table from the record input output area to the screen copy code area.

PFK-PAK-AREA
Set to a value based on the key entered.

VALUE 00	ENTER was pressed
VALUE 01	PF 01 was pressed
VALUE 24	PF 24 was pressed
VALUE 99	PA 1/2 or CLEAR was pressed

NOTE: Data in the screen copy code area is returned only when ENTER or one of the PF keys is pressed. Otherwise, the screen copy code area will contain blank values.

CURSOR POSITIONING - Reading a Screen

- Set to a value based on the position of the cursor on the screen received.
- 1 - N (N is the number of fields on the screen.) This value is only set when ENTER or a PF key is pressed.

CURSOR-POSITIONING - Writing a Screen
When writing a screen, cursor positioning can be set as follows:

1. Placing an absolute value between 1 - N in the CURSOR-POSITION field of the CONTROL-AREA.
   N is the number of fields on the screen. The cursor on the output screen is placed in the field specified by this value. This value refers to the relative field number of the data field in the screen copy code area, not to the relative sequence of the field on the screen itself. The sequence of fields on the screen may be different from the sequence of fields in the screen copy code area.

2. Place a value of zero, (0), in the CURSOR-POSITION field and place the attribute character of E in the cursor attribute for this field in the screen copy code..

   This method allows multiple fields to be set with the cursor attribute during screen processing and allows the screen program to find the first field where the cursor is to be placed. This method is recommended.

It becomes unnecessary to know the relative sequence number of a field in the copy code area.

It becomes unnecessary to keep track of whether a cursor position has been set during screen processing. The screen program will look for a cursor attribute of E whenever the CURSOR-POSITIONING field contains a value of zero. The first field found on the screen (in the sequence of fields on the screen, not in sequence of fields on the screen copy code) is automatically assigned the cursor.

DEFAULT CURSOR POSITIONING - Writing a Screen
The default cursor position is the first blank, unprotected field on the screen.

RETURN-CC
See Appendix E for explanation of codes returned upon completion of the function.

> NOTE: For normal programming it is only necessary to check for RETURN-CC = 0 (the OK condition), or RETURN-CC = 26 (out of bounds), or RETURN-CC = 33 (no data on screen). RETURN-CC should be checked after the EDIT-ERROR-FLAG is checked.

EDIT-ERROR-FLAG
- Set to a value of Y or N.
- Value of Y indicates that errors were encountered during the screen editing.
- Value of N indicates no errors.
- If set to a value of U, upon switch back to **TABLES/AS**, AS will skip the update and move on to the next record.

SEGMENT-TABLE-NAME
Name of the segment or table for which the MAPI or MAPO operation is to be performed.

DBD-NAME
Name of the DBD if the record is an IMS segment for MAPI or MAPO.

MAP-OCC-NO
- A value between 1-N.
- When a given record occurs once on the screen, the value of MAP-OCC-NO should always be 1. A MAP-OCC-NO of zero will return a RETURN-CC of 26, but a MAP-OCC-NO of > 1, will cause the same record to be re-mapped and a RETURN-CC of 0.

- When a record occurs many times on the screen, the value of MAP-OCC-NO should be 1 - N (N is the highest occurrence of the record on the screen). A MAP-OCC-NO = 0 or > N will return a RETURN-CC of 26.

SCREEN-COPY-CODE-AREA

- Contains the data read from the screen in the format of the screen copy code.
- Value of a field is always initialized to spaces in the screen copy code area when no data is entered on the screen in that field.
- Four attribute characters are always initialized to spaces.
- All of the fields previously entered plus those corrected are returned, when errors are encountered in editing and the screen is displayed back to the user and that screen is re-read.
- Data is mapped from this area for MAPI operations.
- Data is mapped to this area for MAPO operations.

LOGICAL-TERMINAL-PCB

- IO-PCB for the application program.
- Used to make IMS GN or GU calls to receive the message.

RECORD-INPUT-OUTPUT-AREA

- This area contains the data in its mapped record form.
- Data is mapped to this area for MAPI operations.
- Data is mapped from this area for MAPO operations.

DATA ABSENCE

A RETURN-CC of 33 indicates that no data was found. This absence of data is interpreted differently when fields occur multiple times. There are four possible situations.

- Each field of the record occurs only once on the screen. Data is considered absent if every field is blank.
- Each field of the record occurs more than once on the screen. Data is considered absent if all fields in a specified occurrence are blank.
- Some fields occur only once on the screen, and some other fields occur many times. Data is considered absent if the fields that occur only once are blank, and the other fields in the specified occurrence are also blank.
- Some fields occur only once on the screen, and some other fields occur many times, but all occurrences are mapped in one operation. Data is considered absent if all participating fields and occurrences are blank.

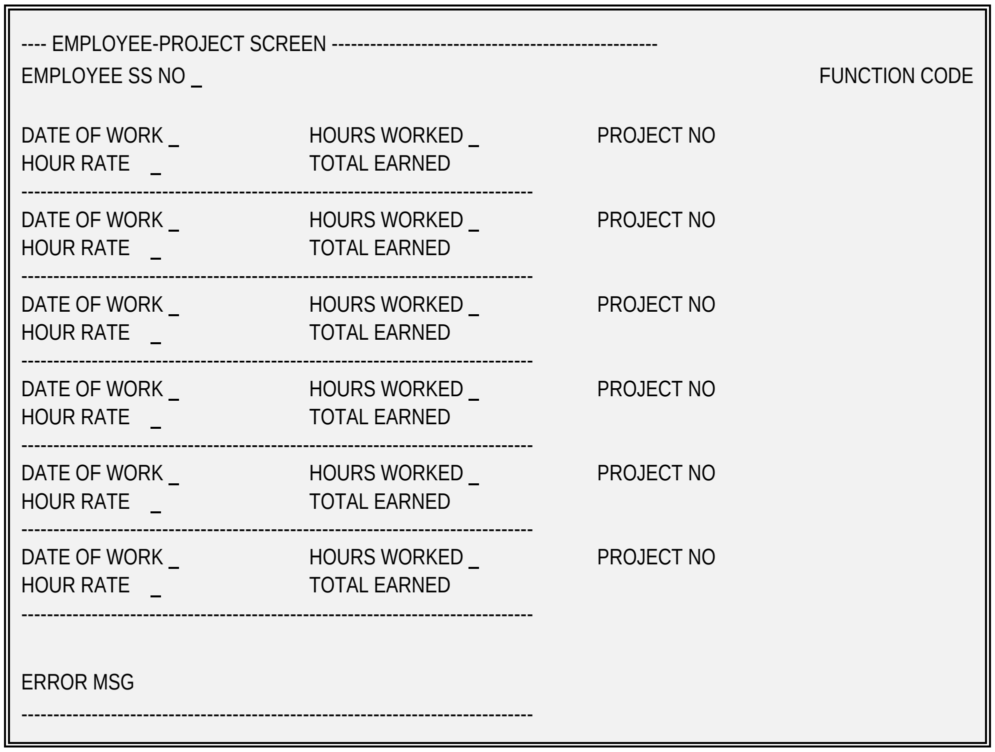
Programming Example

SAMPLE PROGRAM

This program is designed to process the data entered through the EMPLOYEE-PROJECT screen. The program reads the screen, editing of data, mapping of project (hours, one occurrence at a time), and updating of a table called EDTTST (one occurrence at a time). At the end of successful processing, a re-initialized screen is sent back to indicate this condition to the user.

Application Profile
- Employee SS# is entered as: #999-99-9999
- Project hours are edited: 0.50 - 8.00
- Rate of pay is edited: 6.50 - 45.00
- Date is entered as: MMDDYY
- Duplicate records containing the same employee SS#, project #, and date worked are not allowed.
- The only update function allowed is ADD.

Assume that EDITSCRN has the following format:



```
---- EMPLOYEE-PROJECT SCREEN -------------------------------------------------
EMPLOYEE SS NO _                                                    FUNCTION CODE

DATE OF WORK _             HOURS WORKED _            PROJECT NO
HOUR RATE    _             TOTAL EARNED
------------------------------------------------------------------------------
DATE OF WORK _             HOURS WORKED _            PROJECT NO
HOUR RATE    _             TOTAL EARNED
------------------------------------------------------------------------------
DATE OF WORK _             HOURS WORKED _            PROJECT NO
HOUR RATE    _             TOTAL EARNED
------------------------------------------------------------------------------
DATE OF WORK _             HOURS WORKED _            PROJECT NO
HOUR RATE    _             TOTAL EARNED
------------------------------------------------------------------------------
DATE OF WORK _             HOURS WORKED _            PROJECT NO
HOUR RATE    _             TOTAL EARNED
------------------------------------------------------------------------------
DATE OF WORK _             HOURS WORKED _            PROJECT NO
HOUR RATE    _             TOTAL EARNED
------------------------------------------------------------------------------


ERROR MSG
------------------------------------------------------------------------------
```

A CONVERSATIONAL PROGRAM

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. SAMPLE1.
 AUTHOR. SSI.
*
* THIS IS AN EXAMPLE OF A CONVERSATIONAL PROGRAM USING
  TABLES/AS
*
*
 DATE-COMPILED.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-370.
 OBJECT-COMPUTER. IBM-370.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
* SCRATCH PAD AREA
*
  01    SPA.
    02  FILLER      PIC X(6).
    02  SPA-TRANCODE                          PIC X(8).
    02  SPA-DATA                              PIC X(86).
*
* ABEND IDENTIFICATION AREA
*
  01    MISCELLANEOUS-DATA.
    05  ABEND-MSG                             PIC X(50).
    05  ABEND-RESULT                          PIC X(50).
    05  ABEND-CODE                  PIC S9(4) COMP VALUE +2000.
*
* SPA CALL FUNCTION CODES
*
  01    IMS-FUNC-GU                     PIC X(4) VALUE 'GU'.

  01    IMS-FUNC-ISRT                   PIC X(4) VALUE 'ISRT'.
*
* CONTROL AREA
*
  01    CONTROL-AREA.
    05  SCREEN-FUNCTION                       PIC X(4).
```

```
     05  PFK-PAK-AREA                           PIC 9(2).
     05  CURSOR-POSITION           PIC S9(4) COMP.
     05  RETURN-CC                              PIC 99.
     05  EDIT-ERROR-FLAG                        PIC X(1).
         05   RECORD-NAME                         PIC X(8) VALUE 'EDTTST'.
         05   QUAL-NAME                           PIC X(8) VALUE 'SPACE'.
     05  MAP-OCC-NO                             PIC 9(4) COMP.



*
* SCREEN COPY CODE AREA (SCREEN NAME = EDITSCRN)
*
* COPY EDITSCRN.
  01    EDITSCRN.
     05  EDITSCRN-EMPSSNO-ATTR-STD          PIC  X(1).
     05  EDITSCRN-EMPSSNO-ATTR-EXT          PIC  X(1).
     05  EDITSCRN-EMPSSNO-ATTR-COL          PIC  X(1).
     05  EDITSCRN-EMPSSNO-ATTR-CUR          PIC  X(1).
     05  EDITSCRN-EMPSSNO-DATA                     PIC  X(11).
     05  EDITSCRN-EDITFUNC-ATTR-STD         PIC  X(1).
     05  EDITSCRN-EDITFUNC-ATTR-EXT         PIC  X(1).
     05  EDITSCRN-EDITFUNC-ATTR-COL         PIC  X(1).
     05  EDITSCRN-EDITFUNC-ATTR-CUR         PIC  X(1).
     05  EDITSCRN-EDITFUNC-DATA                    PIC  X(1).
     05  EDITSCRN-GROUP001                  OCCURS 6 TIMES.
         10    EDITSCRN-DATEWORK-ATTR-STD  PIC  X(1).
         10    EDITSCRN-DATEWORK-ATTR-EXT  PIC  X(1).
         10    EDITSCRN-DATEWORK-ATTR-COL PIC  X(1).
         10    EDITSCRN-DATEWORK-ATTR-CUR PIC  X(1).
         10    EDITSCRN-DATEWORK-DATA      PIC  X(6).
         10    EDITSCRN-HOURS-ATTR-STD     PIC  X(1).
         10    EDITSCRN-HOURS-ATTR-EXT     PIC  X(1).
         10    EDITSCRN-HOURS-ATTR-COL     PIC  X(1).
         10    EDITSCRN-HOURS-ATTR-CUR     PIC  X(1).
         10    EDITSCRN-HOURS-DATA                PIC  9(5).
         10    EDITSCRN-PROJECT-ATTR-STD   PIC  X(1).
         10    EDITSCRN-PROJECT-ATTR-EXT   PIC  X(1).
         10    EDITSCRN-PROJECT-ATTR-COL   PIC  X(1).
         10    EDITSCRN-PROJECT-ATTR-CUR   PIC  X(1).
         10    EDITSCRN-PROJECT-DATA              PIC  X(6).
         10    EDITSCRN-RATE-ATTR-STD      PIC  X(1).
```

```
          10    EDITSCRN-RATE-ATTR-EXT          PIC  X(1).
          10    EDITSCRN-RATE-ATTR-COL          PIC  X(1).
          10    EDITSCRN-RATE-ATTR-CUR          PIC  X(1).
          10    EDITSCRN-RATE-DATA                   PIC  9(5).
          10    EDITSCRN-EARNED-ATTR-STD        PIC  X(1).
          10    EDITSCRN-EARNED-ATTR-EXT        PIC  X(1).
          10    EDITSCRN-EARNED-ATTR-COL        PIC  X(1).
          10    EDITSCRN-EARNED-ATTR-CUR        PIC  X(1).
          10    EDITSCRN-EARNED-DATA                 PIC  9(6).
     05  EDITSCRN-EDITMSG-ATTR-STD          PIC  X(1).
     05  EDITSCRN-EDITMSG-ATTR-EXT          PIC  X(1).
     05  EDITSCRN-EDITMSG-ATTR-COL          PIC  X(1).
     05  EDITSCRN-EDITMSG-ATTR-CUR          PIC  X(1).
     05  EDITSCRN-EDITMSG-DATA                   PIC  X(73).




*
* TABLES RECORD IO AREA (TABLE NAME = EDTTST)
*
* COPY EDTTST.
  01    EDTTST.
     05  EDTTST-EMPSSNO                       PIC  X(11).
     05  EDTTST-DATEWORK                      PIC S9(06).
     05  EDTTST-HOURS                         PIC S9(04)V99.
     05  EDTTST-PROJECT                       PIC  X(06).
     05  EDTTST-FILLER01                      PIC  X(01).
     05  EDTTST-RATE                          PIC S9(04)V99.
     05  EDTTST-FILLER02                      PIC  X(09).
     05  EDTTST-EARNED                        PIC S9(05)V99.

  LINKAGE SECTION.
  01    LOGICAL-TERMINAL-PCB.
     05  IO-TERMINAL                          PIC X(08).
     05  IO-RESERVE                           PIC X(02).
     05  IO-STATUS                            PIC X(02).
     05  IO-PREFIX                  PIC X(12).
     05  IO-MOD-NAME                          PIC X(08).
     05  IO-USERID                  PIC X(08).
```

```
 PROCEDURE DIVISION.
  ENTRY DLITCBL USING         LOGICAL-TERMINAL-PCB.

 0000-BEGIN.
  CALL 'CBLTDLI' USING     IMS-FUNC-GU
                           LOGICAL-TERMINAL-PCB
                           SPA.
*
* FIRST TIME READ, CHECK FOR LOW-VALUES IN SPA
*
  IF  SPA-DATA = LOW-VALUE
      MOVE SPACE TO SPA-DATA
      GO TO 0100-SEND-INITIAL-SCREEN.
*
* NOT THE FIRST TIME, RECEIVE SCREEN WITH EDIT OPTION
*
  PERFORM 0015-RECEIVE-SCREEN THRU 0015-EXIT.
*
* TERMINATE CONVERSATION IF PF KEY 3 OR 15 WAS PRESSED
*
  IF  PFK-PAK-AREA = (03 OR 15)
      MOVE SPACE TO SPA-TRANCODE
      PERFORM 0022-INSERT-SPA THRU 0022-EXIT
      GO TO 9999-TERMINATE-PROCESSING.




*
* SEND SCREEN BACK AS IS, IF ANY OTHER PF KEY WAS
* PRESSED
*
  IF   PFK-PAK-AREA > 0
       GO TO 9990-SEND-RESPONSE.
*
* RECEIVE SCREEN & CHECK EDIT ERROR
*
  MOVE 0 TO CURSOR-POSITION.
  IF   EDIT-ERROR-FLAG = 'Y'
       GO TO 9990-SEND-RESPONSE
  ELSE
  PERFORM 0800-TABLE-UPDATE THRU 0800-EXIT
  GO TO 0100-SEND-INITIAL-SCREEN.
 0010-SEND-SCREEN.
  MOVE 'ISRT' TO SCREEN-FUNCTION.
  CALL 'EDITSCRN' USING   CONTROL-AREA
                           EDITSCRN
                           LOGICAL-TERMINAL-PCB.
 0010-EXIT.
  EXIT.
 0015-RECEIVE-SCREEN.
  MOVE 'GNE' TO SCREEN-FUNCTION.
  CALL 'EDITSCRN' USING    CONTROL-AREA
                           EDITSCRN
                           LOGICAL-TERMINAL-PCB.
 0015-EXIT.
  EXIT.
 0022-INSERT-SPA.
  CALL 'CBLTDLI' USING     IMS-FUNC-ISRT
                           LOGICAL-TERMINAL-PCB
                           SPA.
 0022-EXIT.
  EXIT.
 0100-SEND-INITIAL-SCREEN.
  MOVE SPACE TO EDITSCRN.
  MOVE 'E' TO EDITSCRN-EMPSSNO-ATTR-CUR.
  GO TO 9990-SEND-RESPONSE.
 0100-EXIT.
  EXIT.
 0800-TABLE-UPDATE.
  MOVE 1 TO MAP-OCC-NO.
 0800-MAP-NEXT-RECORD.
*


* MAP THIS OCCURRENCE FROM SCREEN TO EDTTST RECORD
*
  MOVE 'MAPI' TO SCREEN-FUNCTION.
  MOVE SPACES TO EDTTST.
  CALL 'EDITSCRN' USING    CONTROL-AREA
                           EDITSCRN
                           EDTTST.
```

```
*
* IF THERE IS NO DATA IN THIS OCCURRENCE EXIT
*
    IF  RETURN-CC = 33 GO TO 0800-EXIT.
*
* UPDATE TABLE EDTTST FROM THE DATA JUST MAPPED
* PERFORM CALCULATIONS AND ANY OTHER PROCESSING
* DESIRED FOR THIS TABLE ROW (RECORD).
*
    ADD 1 TO MAP-OCC-NO.
    GO TO 0800-MAP-NEXT-RECORD.
  0800-EXIT.
    EXIT.
  9990-SEND-RESPONSE.
    PERFORM 0022-INSERT-SPA THRU 0022-EXIT.
    PERFORM 0010-SEND-SCREEN THRU 0010-EXIT.
    GO TO 9999-TERMINATE-PROCESSING.
  9999-TERMINATE-PROCESSING.
    GOBACK.
```

CICS Link Format

The general format of the call to the screen program is:

```
EXEC CICS LINK    PROGRAM ('SCRNPROG')
                  COMMAREA (SCREEN-PROGRAM-INTERFACE)
                  LENGTH (4877)
END-EXEC.
```

```
* SCREEN PROGRAM INTERFACE
 01     SCREEN-PROGRAM-INTERFACE.
        05    CONTROL-AREA.
              10  SCREEN-FUNCTION                     PIC X(4).
              10  PFK-PAK-AREA                  PIC 99.
              10  CURSOR-POSITION                     PIC S9(4)COMP.
              10  RETURN-CC                           PIC 99.
              10  EDIT-ERROR-FLAG                     PIC X.
              10  RECORD-NAME                         PIC X(8).
              10  QUAL-NAME                           PIC X(8).
              10  MAP-OCC-NO      PIC S9(4) COMP VALUE +0.
              10  FILLER                              PIC X(48).
        05    SCREEN-COPY-CODE-AREA                   PIC X(2800).
        05    RECORD-MAP-IO-AREA                      PIC X(2000).
* END OF COPY
```

SCREEN-FUNCTION

PUT	=	Insert the message using the screen copy code area for output data.
GET	=	Read the message and map the data into the screen copy code area.
MAPI	=	Map a given occurrence of a file or a table from the screen copy code area to the record input/output area. The previous contents of other fields in the input/output area are not destroyed. It is possible to map into a record area which already contains some data and the MAPI function is called to replace the value in some fields, but not all.
MAPO	=	Map a given occurrence of a file or of a table from the record input/output area to the screen copy code area.

GETE	=	(Get message and edit the data) is used to read message and map the data into the screen copy code area.

PFK-PAK-AREA
Set to a value based on the key entered:

- VALUE 00	ENTER was pressed
- VALUE 01	PF 01 was pressed
- VALUE 24	PF 24 was pressed
- VALUE 99	PA 1/2 or CLEAR was pressed

NOTE: Data in the screen copy code area is returned only when ENTER or one of the PF keys is pressed. Otherwise the screen copy code area will contain blank values.

CURSOR-POSITIONING - Reading a Screen
• Set to a value based on the position of the cursor on the screen received.
• 1 - N (N is the number of fields on the screen). This value is only set when ENTER or

CURSOR POSITIONING - Writing a Screen
When writing a screen, cursor positioning can be set as follows:

1. Placing an absolute value between 1 - N in the CURSOR-POSITION field of the CONTROL-AREA. N is the number of fields on the screen. The cursor on the output screen is placed in the field specified by this value. This value refers to the relative field number of the data field in the screen copy code area, not to the relative sequence of the field on the screen itself. The sequence of fields on the screen may be different from the sequence of fields in the screen copy code area.

2. Place a value of zero, (0), in the CURSOR-POSITION field and place the attribute character of E in the cursor attribute for this field in the screen copy code.

   This method allows multiple fields to be set with the cursor attribute during screen processing and allows the screen program to find the first field where the cursor is to be placed. This method is recommended.

   It becomes unnecessary to know the relative sequence number of a field in the copy code area.

   It becomes unnecessary to keep track of whether a cursor position has been set during screen processing. The screen program will look for a cursor attribute of E, whenever the CURSOR-POSITIONING field contains a value of zero. The first field found on the screen (in the sequence of fields on the screen, not in sequence of fields on the screen copy code) is automatically assigned the cursor.

DEFAULT CURSOR POSITIONING - Writing a screen
The default cursor position is the first blank, unprotected field on the screen.

RETURN-CC
See Appendix E for an explanation of the codes returned upon completion of function.

NOTE: For normal programming it is only necessary to check for RETURN-CC = 0 (the OK condition), or RETURN-CC = 26 (out of bounds), or RETURN-CC = 33 (no data on screen). RETURN-CC should be checked after the EDIT-ERROR-FLAG is checked.

EDIT-ERROR-FLAG
- Set to a value of Y or N.
- Value of Y indicates that errors were encountered during the screen editing.
- Value of N indicates no errors.
- If set to a value of U, upon switch back to **TABLES/AS**, AS will skip the update and move on to the next record.

RECORD-NAME
Name of the record or table for which the MAPI or MAPO operation is to be performed.

MAP-OCC-NO
- A value between 1-N.
- When a given record occurs once on the screen the value of MAP-OCC-No should always be 1. A MAP-OCC-NO of zero will return a RETURN-CC of 26, but a MAP-OCC-NO of >1, will cause the same record to be re-mapped and a RETURN-CC of 0.
- When a record occurs many times on the screen, the value of MAP-OCC-NO should be 1 - N (N is the highest occurrence of the record on the screen). A MAP-OCC-NO = 0 or > N will return a RETURN-CC of 26.

SCREEN-COPY-CODE-AREA
- Contains the data read from the screen in the format of the screen copy code.
- Value of a field is always initialized to spaces in the screen copy code area when no data is entered on the screen in that field.
- Four attribute characters are always initialized to spaces.

- All of the fields previously entered plus those corrected are returned, when errors are encountered in editing and the screen is displayed back to the user and that screen is re-read.
- Data is mapped from this area for MAPI operations.
- Data is mapped to this area for MAPO operations.

RECORD-INPUT-OUTPUT-AREA

- This area contains the data in its mapped record form.
- Data is mapped to this area for MAPI operations.
- Data is mapped from this area for MAPO operations.

DATA ABSENCE

A RETURN-CC of 33 indicates that no data was found. This absence of data is interpreted differently when fields occur multiple times. There are four possible situations.

- Each field of the record occurs only once on the screen. Data is considered absent if every field is blank.
- Each field of the record occurs more than once on the screen. Data is considered absent if all fields in a specified occurrence are blank.
- Some fields occur only once on the screen, and some other fields occur many times. Data is considered absent if the fields that occur only once are blank, and the other fields in the specified occurrence are also blank.
- Some fields occur only once on the screen, and some other fields occur many times, but all occurrences are mapped in one operation. Data is considered absent if all participating fields and occurrences are blank.

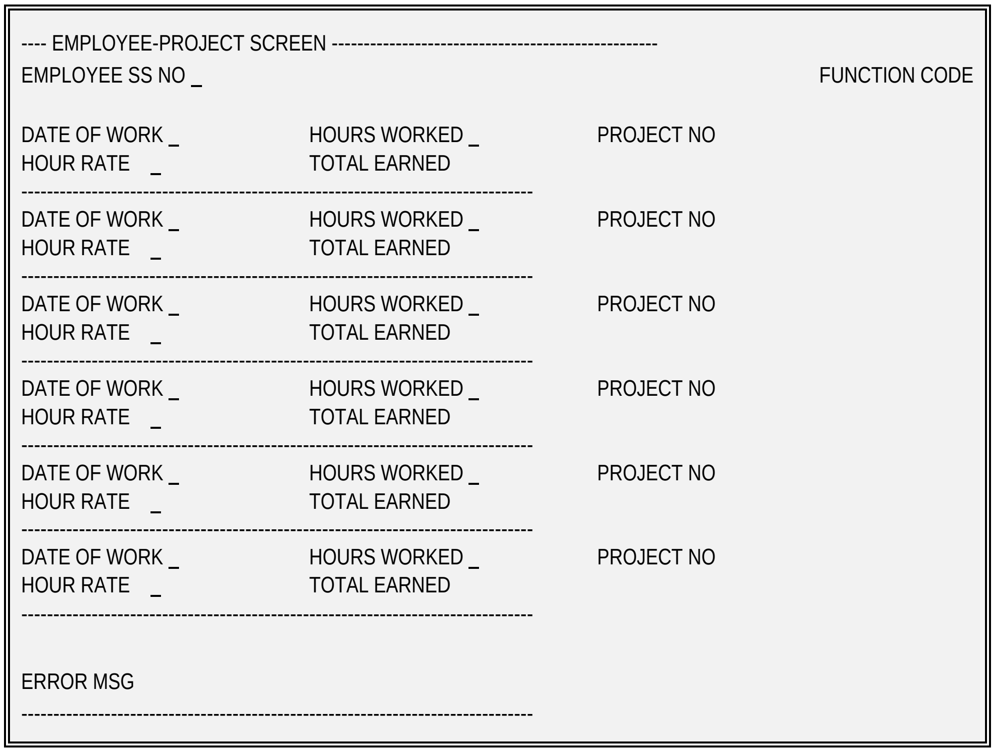
Programming Example

SAMPLE PROGRAM

This program is designed to process the data entered through the EMPLOYEE-PROJECT screen. The program reads the screen, editing of data, mapping of project (hours, one occurrence at a time), and updating of a table called EDTTST (one occurrence at a time). At the end of successful processing, a re-initialized screen is sent back to indicate this condition to the user.

Application Profile
• Employee SS# is entered as: #999-99-9999
• Project hours are edited: 0.50 - 8.00
• Rate of pay is edited: 6.50 - 45.00
• Date is entered as: MMDDYY
• Duplicate records containing the same employee SS#, project #, and date worked are not allowed.
• The only update function allowed is ADD.

Assume that EDITSCRN has the following format:



```
---- EMPLOYEE-PROJECT SCREEN --------------------------------------------------
EMPLOYEE SS NO _                                                    FUNCTION CODE

DATE OF WORK _            HOURS WORKED _            PROJECT NO
HOUR RATE    _            TOTAL EARNED
--------------------------------------------------------------------------------
DATE OF WORK _            HOURS WORKED _            PROJECT NO
HOUR RATE    _            TOTAL EARNED
--------------------------------------------------------------------------------
DATE OF WORK _            HOURS WORKED _            PROJECT NO
HOUR RATE    _            TOTAL EARNED
--------------------------------------------------------------------------------
DATE OF WORK _            HOURS WORKED _            PROJECT NO
HOUR RATE    _            TOTAL EARNED
--------------------------------------------------------------------------------
DATE OF WORK _            HOURS WORKED _            PROJECT NO
HOUR RATE    _            TOTAL EARNED
--------------------------------------------------------------------------------
DATE OF WORK _            HOURS WORKED _            PROJECT NO
HOUR RATE    _            TOTAL EARNED
--------------------------------------------------------------------------------


ERROR MSG
--------------------------------------------------------------------------------
```

SAMPLE PROGRAM

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SAMPLE.
REMARKS.
THIS IS AN EXAMPLE OF A PROGRAM WRITTEN TO USE TABLES/AS
SCREEN PROGRAM.
DATA DIVISION.
WORKING-STORAGE SECTION.
*
* SCREEN PROGRAM INTERFACE (COPY MEMBER "SPICOPY")
*
 01   SCREEN-PROGRAM-INTERFACE.
  05  CONTROL-AREA.
      10  SCREEN-FUNCTION                          PIC X(4).
      10  PFK-PAK-AREA                             PIC 99.
      10  CURSOR-POSITION                          PIC S9(4) COMP.
      10  RETURN-CC                                PIC 99.
      10  EDIT-ERROR-FLAG                          PIC X.
      10  RECORD-NAME                              PIC X(8).
      10  QUAL-NAME                                PIC X(8).
      10  MAP-OCC-NO      PIC S9(4) COMP VALUE +0.
      10  FILLER                               PIC X(48).
  05  SCREEN-COPY-CODE-AREA                        PIC X(2800).
  05  RECORD-MAP-IO-AREA                           PIC X(2000).
* END OF COPY

 01   EDITSCRN.
  05  EDITSCRN-EMPSSNO-ATTR-STD            PIC  X(1).
  05  EDITSCRN-EMPSSNO-ATTR-EXT            PIC  X(1).
  05  EDITSCRN-EMPSSNO-ATTR-COL            PIC  X(1).
  05  EDITSCRN-EMPSSNO-ATTR-CUR            PIC  X(1).
  05  EDITSCRN-EMPSSNO-DATA                    PIC  X(11).
  05  EDITSCRN-EDITFUNC-ATTR-STD           PIC  X(1).
  05  EDITSCRN-EDITFUNC-ATTR-EXT           PIC  X(1).
  05  EDITSCRN-EDITFUNC-ATTR-COL           PIC  X(1).
  05  EDITSCRN-EDITFUNC-ATTR-CUR           PIC  X(1).
  05  EDITSCRN-EDITFUNC-DATA                   PIC  X(1).
  05  EDITSCRN-GROUP001                        OCCURS 6 TIMES.
      10    EDITSCRN-DATEWORK-ATTR-STD  PIC  X(1).
      10    EDITSCRN-DATEWORK-ATTR-EXT  PIC  X(1).
```

```
          10    EDITSCRN-DATEWORK-ATTR-COL PIC  X(1).
          10    EDITSCRN-DATEWORK-ATTR-CUR PIC  X(1).
          10    EDITSCRN-DATEWORK-DATA     PIC  X(6).
          10    EDITSCRN-HOURS-ATTR-STD    PIC  X(1).
          10    EDITSCRN-HOURS-ATTR-EXT    PIC  X(1).
          10    EDITSCRN-HOURS-ATTR-COL    PIC  X(1).
          10    EDITSCRN-HOURS-ATTR-CUR    PIC  X(1).
          10    EDITSCRN-HOURS-DATA              PIC  9(5).
          10    EDITSCRN-PROJECT-ATTR-STD  PIC  X(1).
          10    EDITSCRN-PROJECT-ATTR-EXT  PIC  X(1).
          10    EDITSCRN-PROJECT-ATTR-COL  PIC  X(1).
          10    EDITSCRN-PROJECT-ATTR-CUR  PIC  X(1).
          10    EDITSCRN-PROJECT-DATA            PIC  X(6).
          10    EDITSCRN-RATE-ATTR-STD     PIC  X(1).
          10    EDITSCRN-RATE-ATTR-EXT     PIC  X(1).
          10    EDITSCRN-RATE-ATTR-COL     PIC  X(1).
          10    EDITSCRN-RATE-ATTR-CUR     PIC  X(1).
          10    EDITSCRN-RATE-DATA               PIC  9(5).
          10    EDITSCRN-EARNED-ATTR-STD   PIC  X(1).
          10    EDITSCRN-EARNED-ATTR-EXT   PIC  X(1).
          10    EDITSCRN-EARNED-ATTR-COL   PIC  X(1).
          10    EDITSCRN-EARNED-ATTR-CUR   PIC  X(1).
          10    EDITSCRN-EARNED-DATA             PIC  9(6).
       05 EDITSCRN-EDITMSG-ATTR-STD        PIC  X(1).
       05 EDITSCRN-EDITMSG-ATTR-EXT        PIC  X(1).
       05 EDITSCRN-EDITMSG-ATTR-COL        PIC  X(1).
       05 EDITSCRN-EDITMSG-ATTR-CUR        PIC  X(1).
       05 EDITSCRN-EDITMSG-DATA                  PIC  X(73).

      01  EDTTST.
       05 EDTTST-EMPSSNO                         PIC  X(11).
       05 EDTTST-DATEWORK                        PIC S9(06).
       05 EDTTST-HOURS                           PIC S9(04)V99.
       05 EDTTST-PROJECT                         PIC  X(06).
       05 EDTTST-FILLER01                        PIC  X(01).
       05 EDTTST-RATE                            PIC S9(04)V99.
       05 EDTTST-FILLER02                        PIC  X(09).
       05 EDTTST-EARNED                          PIC S9(05)V99.

      01  DUMMY-COMMAREA                         PIC X.

      01  END-MESSAGE                            PIC X(40)
```

```
           VALUE 'TRANSACTION TERMINATED'.
       LINKAGE SECTION.
       01  DFHCOMMAREA                            PIC X.
       PROCEDURE DIVISION.
        IF EIBCALEN EQUAL ZERO GO TO
           0100-SEND-INITIAL-SCREEN.
        MOVE 'GETE' TO SCREEN-FUNCTION.
        PERFORM 0020-CALL-TABLE-AS THRU 0020-EXIT.
        MOVE SCREEN-COPY-CODE-AREA TO EDITSCRN.
       * TERMINATE TRANSACTION IF PF3 OR PF15 WAS PRESSED
        IF PFK-PAK-AREA EQUAL (03 OR 15)
        GO TO 9999-END-TRANSACTION.
       * RE-SEND SCREEN IF ANY KEY OTHER THAN ENTER WAS PRESSED
        IF PFK-PAK-AREA GREATER THAN ZERO
        GO TO 9990-SEND-RESPONSE.
       * CHECK EDIT ERROR
        MOVE 0 TO CURSOR-POSITION.
        IF EDIT-ERROR-FLAG EQUAL 'Y' GO TO
           9990-SEND-RESPONSE
        ELSE
        PERFORM 0800-TABLE-UPDATE THRU 0800-EXIT
        GO TO 0100-SEND-INITIAL-SCREEN.
       0020-CALL-TABLE-AS.
        EXEC CICS LINK                          PROGRAM ('EDITSCRN')
                   COMMAREA (SCREEN-PROGRAM-INTERFACE)
                   LENGTH (4877)
        END-EXEC.
       0020-EXIT.
        EXIT.
       0100-SEND-INITIAL-SCREEN.
        MOVE SPACE TO EDITSCRN.
        MOVE 'E' TO EDITSCRN-EMPSSNO-ATTR-CUR.
        GO TO 9990-SEND-RESPONSE.
       0100-EXIT.
        EXIT.
       0800-TABLE-UPDATE.
        MOVE 1 TO MAP-OCC-NO.
       0800-MAP-NEXT-RECORD.
      *
      * MAP THIS OCCURRENCE FROM SCREEN TO EDTTST RECORD
      *
        MOVE 'MAPI' TO SCREEN-FUNCTION.
```

```
    MOVE SPACES TO RECORD-MAP-IO-AREA.
    PERFORM 0020-CALL-TABLES-AS THRU 0020-EXIT.
    MOVE RECORD-MAP-IO-AREA TO EDTTST.
*
* IF THERE IS NO DATA IN THIS OCCURRENCE EXIT.
*
   IF  RETURN-CC = 33 GO TO 0800-EXIT.
*
* UPDATE TABLE EDTTST FROM THE DATA JUST MAPPED.
* PERFORM CALCULATIONS AND ANY OTHER PROCESSING DESIRED *
FOR THIS TABLE ROW (RECORD).
*
    ADD 1 TO MAP-OCC-NO.
    GO TO 0800-MAP-NEXT-RECORD.
  0800-EXIT.
    EXIT.
  9990-SEND-RESPONSE.
    MOVE EDITSCRN TO SCREEN-COPY-CODE-AREA.
    MOVE 'PUT' TO SCREEN-FUNCTION.
    PERFORM 0020-CALL-TABLE-AS THRU 0020 EXIT.
  9999-TERMINATE-PROCESSING.
    EXEC CICS RETURN   TRANSID ('SAMP')
                  COMMAREA (DUMMY-COMMAREA)
                  LENGTH (1)
    END-EXEC.
    GOBACK.
```

Section 7

TABLES/AS Technical Guide

T S L U

Memory Resident

Table Processing

Transaction

T S L U Memory Resident Table Processing Transaction

Memory Resident Table Processing Transaction - TSLU

TSLU - Only supported in IMS and CICS, not TSO

This transaction may be executed to perform any memory resident table processing and provide pre-load statistics. To enter new commands, press enter key. To terminate this transaction enter clear key or any of the PA keys.

FORMAT OF INPUT
Input to this transaction is entered on the terminal beginning at column 1.

COLUMN 1-4
TSLU or its equivalent transaction code assigned.

COLUMN 6-9
A function code among the following:

LOAD	Load a given table (must issue a FREE call to re-load a table) To have the table loaded in ascending sequence for a column, enter the column name following one space after the table name.
FREE	Free a given table
GETF	Retrieve first record from a table.
GETN	Retrieve the next record from the given table.
STGF	Displays statistics on the given table name: • No of records loaded • Length of a table row • Amount of memory being used • Number of accesses made since loading • Date and time the table was loaded • Userid which initiated the load

```
Format of Display:
TSLU STGF SCRSWC
FUNCTION: STGF     TABLE: SCRSWC     RESULT: OK
TABLE NAME:                 SCRSWC
ADDRESS:                  02570960*
TABLE SIZE:                   1,680
ROWS:                            67
COLUMNS:                         14
ROW LENGTH:                      98
ACCESSES:                         0
LOADED BY:                   SPS001
LOAD TIME STAMP: 1990-05-09-09.43.21.984278*
```

* ADDRESS: Is displayed in hexadecimal characters
LOAD TIME STAMP: Time stamp when the table loading was completed.

If all 0's, the table is loaded in a Dataspace.

STGN Displays statistics on the next table pre-loaded. Table name is not required. This function not valid in IMS.

COLUMN 11-46
DB2 table name up to 36 characters fully qualified name.

T S L U Memory Resident Table Processing Transaction

CICS Sequential Terminal Input for Preloading Tables

To preload tables in CICS, a sequential terminal can be used to execute transaction TSLU. The following entries must be placed in the TCT, in the order shown:

```
COLS
0    1     1                                                      7
1----0-----6------------------------------------------------------2
SEQ1 DFHTCT TYPE=SDSCI,                                 X
        DEVICE=DASD,                            X
        DSCNAME-SSISEQI,MACRF=R
    DFHTCT TYPE=SDSCI,                                X
        DEVICE-DASD,                            X
        DSCNAME-SSISEQ0,MACRF=W
    DFHTCT TYPE=LINE,ACCMETH=SAM,INAREAL-800,TRMTYPE=DASD,   X
        ISADSCN-SSISEQI,                          X
        OSADSCN-SSISEQ0
    DFHTCT TYPE=TERMINAL,                               X
        TRMIDNT-SEQ1
|----|-----|----------------------------------------------------|
```

The DSCNAME on the two TYPE=SDSCI statements can be any standard names but must match the ISADSCN and OSADSCN values on the TYPE=LINE statement. Also, the DSCNAMES are the DD names of the files that must be in the CICS startup JCL. These DD statements can be as follows:

```
//SSISEQI   DD      DSN=SEQ.TERM.INPUT,DISP=SHR
//SSISEQO  DD      SYSOUT=A,DCB=(BLKSIZE=800,RECFM=F,LRECL=800)
```

Where SSISEQI must be: A standard sequential dataset with DCB of BLKSIZE=80=,RECFM=F,LRECL=-80
and SSISEQO should be as shown. It could be a sequential dataset with the DCB as shown.

To preload tables, edit the SEQ.TERM.INPUT dataset and specify the transactions to be executed. The TABLES transaction TSLU is used to preload tables. See the following example for format:

```
COLS
1-----------------------------------------------------------------

TSLU LOAD SSI.DB2TEST \              ===> Load a DB2 Table
TSLU LOAD VSAM.ACCTAB \              ===> Load a VSAM File
TSLU END \                           ===> Req'd to End TSLU Transaction
CSSF GOODNIGHT \                 ===> Req'd to End Input
```

T S L U Memory Resident Table Processing Transaction

1---

The TSLU LOAD transactions cause the named tables to be loaded. The TSLU END transaction is req'd after the last TSLU LOAD transaction to end the preceding TSLU conversational transaction. The CSSF GOODNIGHT transaction is req'd to end the input, otherwise an error will occur. The ' \ ' is req'd to mark the end of a transaction. Anything after the backslash is considered a comment and is ignored.

Technical Guide

Appendices

Appendix A
Date Format Codes

1	MMDDYY	
2	MMDDYYYY	
3	DDMMYY	
4	DDMMYYYY	
5	YYMMDD	
6	YYYYMMDD	
7	MM/DD/YY	
8	MM/DD/YYYY	
9	DD/MM/YY	
A	DD/MM/YYYY	
B	YY/MM/DD	
C	YYYY/MM/DD	
D	YYDDD	
E	YYYYDDD	
F	YY/DDD	
G	YYYY/DDD	
H	DD-MMM-YY	(`01-JAN-90',L=9)
I	DD-MMM-YYYY	(`01-JAN-1990',L=11)
J	MON DD,YYYY	(`JUNE 1, 1990',L=18)
K	MMM DD,YYYY	(`OCT 1, 1990',L=12)
L	YYYY-MM-DD	(DB2 FORMAT, ISO, JAPANESE)
M	DD.MM.YYYY	(EUROPEAN STD)
S	TIME STAMP	(26 CHARACTER FORMAT)

Appendix B
Field Display Attributes

STANDARD ATTRIBUTE CHARACTERS

A	Input/output field, low intensity
B	Input/output field, high intensity
C	Input/output field, non-displayable
D	Output field, low intensity
E	Output field, high intensity
F	Output field, non-displayable
G	Input/output protected field, low intensity, modified data tag on
H	Input/output protected field, high intensity, modified data tag on
I	Input/output protected field, non-displayable, modified data tag on (used to hide fields on the screen)

EXTENDED ATTRIBUTE CHARACTERS

B	Blink
R	Reverse video
U	Underline
Blank	Device Default

COLOR ATTRIBUTE CHARACTERS

B	Blue
R	Red
P	Pink
G	Green
T	Turquoise
Y	Yellow
N	Neutral
Blank	Device Default

CURSOR ATTRIBUTE CHARACTERS

E	Position cursor on this field
Blank	Do not position cursor on this field

Appendix C
TABLES/AS Screen Processor Control

SEARCH/COMPARE OPERATORS

<	Less than
>	Greater than
=	Equal to
L	Less than or equal to
G	Greater than or equal to
+	Not equal to
J	Not like (DB2 interface only)
K	Like (DB2 interface only)

FUNCTION CODES

R, B, I	Retrieve for inquire only
U	Retrieve for update (This is the default if left blank.)
A	Add
C	Change (also, blank with any data changed in the row)
D	Delete

Appendix D
Editor Function Codes

FUNCTION CODES PASSED TO THE EDITOR PROGRAMS

• Value passed during record processing.

RB	=	Retrieve for inquiry (before retrieval - once)
RA	=	Retrieve for inquiry (after retrieval - once per row)
UB	=	Retrieve for update (before retrieval - once)
UA	=	Retrieve for update (after retrieval -once per row)
EA	=	Edit add data prior to validation process (once per row)
EC	=	Edit change data prior to validation process (once per row)
ED	=	Edit delete data prior to validation process (once per row)
AB	=	Table add process (before add - once per row)
AA	=	Table add process (after add - once per row)
DB	=	Table delete process (before delete - once per row)
DA	=	Table delete process (after delete - once per row)
CB	=	Table change process (before change - once per row)
CA	=	Table change process (after change - once per row)
SI	=	Just after Screen Input (EDIT-ERROR-FLAG may be set to `Y' (once)
SO	=	Just before Screen Output (once)

• Values passed during screen switching.

F	=	Switch from screen (There is no data in the 2 I/O areas, only in the screen copy code area.)
T	=	Switch to screen (There is no data in the 2 I/O areas, only in the screen copy code area.)

• Value passed when editor is controlling the processing flow

Blank	=	To be processed via the editor mode (SCREDT entry)

Appendix E
Screen Programming Processor Interface Return Codes

	GETE	GN	GNE	GU	GUE	IRST	MAPI	MAPO	EXPLANATION
00	X	X	X	X	X	X	X	X	OK - Function completed successfully.
01	X	X	X	X	X	X	X	X	Unrecognizable function code.
02		X		X					Invalid status trying to GN or GU the message from IMS. Check the IO-PCB for the IMS status code. Invalid status trying to GET the message from CICS.
03						X			Invalid status trying to ISRT (insert) the message to IMS. Check the IO-PCB for the IMS status code. Invalid status trying to PUT the message to CICS.
10	X		X		X				Editor returned an error flag of Y on function code GNE, GUE or GETE. (Since the EDIT-ERROR-FLAG is also set to Y, it is only necessary to check the EDIT-ERROR-FLAG for function codes GNE, GUE and GETE).
13	X		X		X				No editing was defined, but the function code used was GNE, GUE or GETE.
17							X	X	No mapping was defined, but function code used was MAPI or MAPO.
20						X			Attempted a second/double insert of a screen.
24	X	X	X	X	X				Attempted to retrieve/get a screen without having previously inserted it.
26							X	X	For the MAPI or MAPO function, the MAP-OCC-NO field in the CONTROL-AREA contains a value greater than the number of occurrences on the screen. (Out of bounds condition.)
27							X	X	Unrecognizable record field type. (Not = C, P , B, or D). If this condition occurs, contact Specialized Software International, Inc. customer service.

	GETE	GN	GNE	GU	GUE	IRST	MAPI	MAPO	EXPLANATION
28							X		Invalid mapping of an alphanumeric format field to a numeric field in the record. If this condition occurs, contact Specialized Software Internal, Inc. customer service.
29							X	X	The call MAPI or MAPO requires a table name or a segment and DBD name or a copy code name. This error indicates that it did not find a match in one of these names.
30							X	X	The mapping operation requested is invalid because the field on the screen is not compatible to this corresponding field in the record. For example, an alphanumeric without format to a numeric field. If this condition occurs, contact Specialized Software International, Inc. customer service.
31								X	A field of zero length was detected while mapping was being performed. The mapping operation was suspended. If this condition occurs, contact Specialized Software International, Inc. customer service.
32							X		A quantity field being mapped was found to contain an erroneous definition for screen input. The mapping operation was suspended. If this condition occurs, contact Specialized Software International, Inc. customer service.
33							X		All of the fields within the specified map occurrence contained blank characters. This condition indicates that there is no data in any of the fields participating in this mapping operation.
34							X	X	The map program has found an error in the date output format code. If this condition occurs, contact Specialized Software International, Inc. customer service.
35							X	X	The map program has found an error in the date input format code. If this condition occurs, contact Specialized Software International, Inc. customer service.

Appendix F
TABLES/AS Control Tables

The following Control Tables are used in the **TABLES/AS** system to store screen related information for processing. They should not be updated unless specified by Specialized Software, except for AS_Messages. This table is provided for your use.

	APPLID	Application ID's assigned
	AUDITDB	Log Records
P	**AS_MESSAGES**	Global Message Table
	IMSSE2	IMS database segment control information
P	**IMSSEG**	Segment search arguments
	LAYOUT	Record layout of COBOL or PL1 copy member
P	**MS_DEFINITION**	Effectivity control and Order-By definitions
P	**SCPMSG**	User specified messages for the Screen Processor
	SCRCHK	Relational edit conditions
P	**SCRCON**	Fields being mapped to the copy code layout
	SCRDFT	Default screen name assignment
P	**SCREDT**	Specifies a user edit program for a screen
	SCREENDB	Screen layout and composite records
	SCRERR	Custom error messages
P	**SCRINT**	Initial values for a screen
	SCRLOK	Protect screens from unauthorized changes
	SCRMSG	Common message field definitions by screen
P	**SCRORD**	DB2 field order by information by screen name
P	**SCRRCD**	Database records defined for processing
P	**SCRSWC**	Screen switching control
	SCRVAL	Edits and Discrete values for a field on a given screen
	TXNTBL	User assigned transaction names
P	**TXNMNU**	Transaction code to user screen switching
P	**VALDTE**	Record level validation to be performed
P	**USERID**	Maintains a list of all user ID's for non-DB2 tables
	VSMSE2	VSAM file control information
	WORKDB1	**TABLES/AS** intermediate work areas
	WORKDB2	**TABLES/AS** intermediate work areas

P indicates pre-loaded tables.

Appendix G
Screen Switching (IMS)

Screen switching involves transferring from one screen to another screen. This transfer may involve:

- Transferring data from one screen to the other
- Saving the data on the screen being transferred FROM
- Restoring data on the screen being transferred TO. The data must have been saved previously.

The following write-ups cover the capabilities and coding requirements for:

- Switching TO and FROM applications which use **TABLES/AS** screens.
- Switching TO and FROM applications which use MFS screens.

Switching to and from Applications Which Use TABLES/AS Screens

The following types of screen switching can be defined for applications using **TABLES/AS** screens:

1. From the application program to the **TABLES/AS** Screen Processor menu.

2. From the application program to a **TABLES/AS** screen within Screen Processor.

3. From a **TABLES/AS** screen within Screen Processor to an application program
   A. Using a PF key (with or without data transfer).
   B. Using an option code (with or without data transfer).

In order to switch to an application program using **TABLES/AS** screens from the Screen Processor, the application must be conversational.

1. Switching from an Application Program Using TABLES/AS Screens to Screen Processor Menu

In order to switch from the application program to the **TABLES/AS** Screen Processor menu, a SPA (scratch pad area) size of at least 100 bytes must be communicated to the Screen Processor in the following format:

```
01    SPA
      05   FILLER                        PIC X(6).
      05   TRAN-CODE                     PIC X(8) VALUE
                    'TS09'. OR ITS EQUIVALENT
      05   FILLER                        PIC X(8).
      05   COND-CODE                     PIC X(1) VALUE ZERO.
      05   INITIATION-SOURCE             PIC X(1) VALUE 'P'.
      05   FILLER                        PIC X(24).
      05   ACCESS-ALLOWED                PIC X(4) VALUE
                                                  LOW-VALUES.
      05   FILLER                        PIC X(48).
```

After the SPA is inserted, issue the following call:

```
        CALL 'TB2MAINT' USING       CONTROL AREA
                                    SCREEN-COPY-CODE-AREA
                                    LOGICAL-TERMINAL-PCB.
```

The SCREEN-FUNCTION in the CONTROL-AREA must be set to ISRT.

If cursor attribute, TB2MAINT-TB2SCRN-ATTR-CUR, is set in the SCREEN-COPY-CODE-AREA, the CURSOR-POSITION in CONTROL-AREA must be set to zero. Otherwise, it should be set to one or two.

The SCREEN-COPY-CODE-AREA is the copy code of the **TABLES/AS** Screen Processor menu. It may be initialized to spaces or a screen name and/or message may be passed in order to be displayed on the Screen Processor menu.

The record format of the CONTROL-AREA is:

```
   01    CONTROL-AREA.
         05    SCREEN-FUNCTION            PIC X(4).
         05    PFK-PAK-AREA               PIC 9(2).
         05    CURSOR-POSITION            PIC S9(4) COMP.
         05    RETURN-CC                  PIC 99.
         05    EDIT-ERROR-FLAG            PIC X(1).
         05    RECORD-NAME                PIC X(8).
         05    QUAL-NAME                  PIC X(8).
         05    MAP-OCC-NO                 PIC 9(4) COMP.
         05    SCREEN-PROGRAM-NAME        PIC X(8).
         05    TRANSACTION-CODE           PIC X(8).
```

The record format of SCREEN-COPY-CODE-AREA for the Screen Processor menu is:

```
 01   TB2MAINT
      05    TB2MAINT-TB2SCRN-ATTR-STD              PIC X(1).
      05    TB2MAINT-TB2SCRN-ATTR-EXT              PIC X(1).
      05    TB2MAINT-TB2SCRN-ATTR-COL              PIC X(1).
      05    TB2MAINT-TB2SCRN-ATTR-CUR              PIC X(1).
      05    TB2MAINT-TB2SCRN-DATA                          PIC X(8).
      05    FILLER                                 PIC X(36).
      05    TB2MAINT-TB2MESAG-ATTR-STD             PIC X(1).
      05    TB2MAINT-TB2MESAG-ATTR-EXT             PIC X(1).
      05    TB2MAINT-TB2MESAG-ATTR-COL             PIC X(1).
      05    TB2MAINT-TB2MESAG-ATTR-CUR             PIC X(1).
      05    TB2MAINT-TB2MESAG-DATA                 PIC X(79).
WHERE,
 TB2MAINT-TB2SCRN-DATA  = TABLES/AS SCREEN NAME TO
```

```
BE PROCESSED TB2MAINT-TB2MESAG-DATA = ANY DISPLAY
MESSAGE
```

The record format of LOGICAL-TERMINAL-PCB is:

```
01     LOGICAL-TERMINAL-PCB.
       05     IO-TERMINAL                  PIC X(8).
       05     IO-RESERVE                   PIC X(2).
       05     IO-STATUS                    PIC X(2).
       05     IO-PREFIX                    PIC X(12).
       05     IO-MOD-NAME                  PIC X(8).
       05     IO-USERID                    PIC X(8).
```

When switching between the screen processor to an Application Program and the data currently shown on the application screen is to be redisplayed upon return, do the following:

a. In your program, write a record using SQL insert into the TABLES control table WORKDB1 using the following instructions:

```
01  TABLE-IO-AREA.
    05    IO-AREA-SIZE         PIC S9(4) COMP VALUE
                                +3027.
    05    TABLE-RECORD-KEY.
          10  IO-TERMINAL-ID   PIC X(8).
          10  IO-SCREEN-NAME   PIC X(13).
          10  IO-DATA-TYPE     PIC X(1) VALUE 'B'.
          10  IO-SEGMENT-NO    PIC S9(4) COMP VALUE
                                +1.
    05    IO-SEGMENT-ID        PIC X VALUE '1'.
    05    IO-SCREEN-DATA       PIC X(3000).

WHERE,
    IO-TERMINAL-ID = LOGICAL TERMINAL ID
    IO-SCREEN-NAME = NAME OF THE SCREEN TO BE
                  RESTORED, MOD NAME FOR AN MFS SCREEN
    IO-SCREEN-DATA = SCREEN DATA INCLUDING THE LL
                  & ZZ FIELDS FOR AN MFS SCREEN
```

b. A switching record must be defined to switch back to the application program using the Restore option. See 3A and 3B defined later.

2. Switching from an Application Program Using TABLES/AS Screens to a TABLES/AS Screen within Screen Processor

In order to switch from the application program to **TABLES/AS** screen in the Screen Processor, a SPA (scratch pad area) size of at least 100 bytes must be communicated to the Screen processor in the following format:

```
01     SPA
05     FILLERPIC X(6)
05     TRAN-CODE                       PIC X(8)  VALUE
              'TS09' OR ITS
              EQUIVALENT
05     SCREEN-ID                       PIC X(8)  MUST BE SET
             TO THE TABLES/AS SCREEN
              BEING SWITCHED TO.
05     COND-CODE                       PIC X(1)  VALUE 'X'
05     INITIATION-SOURCE               PIC X(1)  VALUE 'P'
05     RECORD-NAME                     PIC X(8)  (SEE
                                       DISCUSSION BELOW)
05     FILLER                          PIC X(16)
05     ACCESS-ALLOWED                  PIC X(4)  VALUE
                                       LOW-VALUES.
05     FILLERPIC X(48) VALUE
                                       LOW-VALUES.
```

RECORD-NAME
If the TO screen has more than one map but your application is only interested in processing a particular map, enter the map name to be processed from the TO screen.

If your application must utilize more than one map defined on the TO screen, leave this field blank. Response time will be slower since **TABLES/AS** will need to initialize its information for all maps on the TO screen.

If the TO screen has only one map defined, enter the map name. This will improve response time. The map name entered must be a table map or an IMS segment name.

After the SPA is inserted, issue the following call:

```
CALL SCREEN-ID USING  CONTROL AREA
                      SCREEN-COPY-CODE-AREA
                      LOGICAL-TERMINAL-PCB.
```

SCREEN-ID is the TO screen name.

The SCREEN-FUNCTION in the CONTROL-AREA must be set to ISRT.

The CURSOR-POSITION in the CONTROL-AREA may be established based on **TABLES/AS** programming results.

The SCREEN-COPY-CODE-AREA is the copy code of the target screen. It may be initialized to spaces if a blank screen is to be displayed, or it can be pre-set with values if the data is to be passed from the application to the **TABLES/AS** screen.

If switching from the Screen Processor back to the application program and the data currently shown on the application screen is to be redisplayed upon return, you must save your screen on **TABLES** intermediate database using the previous example.

3A. Switching from a TABLES/AS Screen within Screen Processor to an Application Program Using TABLES/AS Screens (using a PF Key)

Obtain the switching screen in **TABLES/AS**, i.e., Opt 7/Opt 3.

Enter the following required fields:

FROM SCREEN
Enter a valid **TABLES/AS** screen name.

TO SCREEN
Enter the name of the **TABLES/AS** screen in the application being switched to.

PF KEY
Enter the PF key number which is to cause the switch. Valid PF keys are 1-11, and 13-24. PF key 12 is reserved.

TRANSACTION CODE
Enter the transaction code of the conversational program being switched to.

OPTIONAL FIELDS

FIELD NUMBER FOR CURSOR POSITIONING

Enter the field number of the field the cursor is to be positioned at on the target screen.

EDITOR PROGRAM NAME
To call a user-written program before exiting the FROM screen and before the TO screen is displayed, enter the editor program name.

TRANSFERRING DATA

MAP NAME TO TRANSFER
Enter the table map, or IMS segment name of the record being transferred.

The table, map or segment record being transferred must be defined as a record map on the FROM screen and the TO screen. Only the common data fields will be transferred. If data is not to be passed from one screen to the other, leave the above two fields blank.

The fields on the screen defined in the record maps will be displayed on the TO screen when the switch occurs.

The Screen Processor will transfer the following SPA to the application program:

```
01  SPA
    05  FILLER    PIC X(6)
    05  TRAN-CODE                              PIC X(8) THE
       TRANSACTION CODE OF THE APPLICATION
       PROGRAM
    05  SCREEN-ID                              PIC X(8) THE NAME OF
                  THE SOURCE SCREEN
    05  COND-CODE                              PIC X(1)
    05  INITIATION-SOURCE                      PIC X(1)
    05  FILLER    PIC X(76) VALUE
                  LOW-VALUES
```

SAVING AND RESTORING DATA
- Enter an S to save data fields on the FROM screen before switching.
- Enter an R to restore data fields previously saved for the TO screen when switching occurs.
- Enter a B to both save and restore.
- Leave the field blank if saving or restoring is not desired.

3B. Switching from a TABLES/AS Screen within Screen Processor to an Application Program Using TABLES/AS Screens (Using an Option Code)

Obtain the switching screen in **TABLE/AS**, i.e., Opt 7/Opt 3.

Enter the following required fields:

FROM SCREEN
Enter a valid **TABLES/AS** screen name.

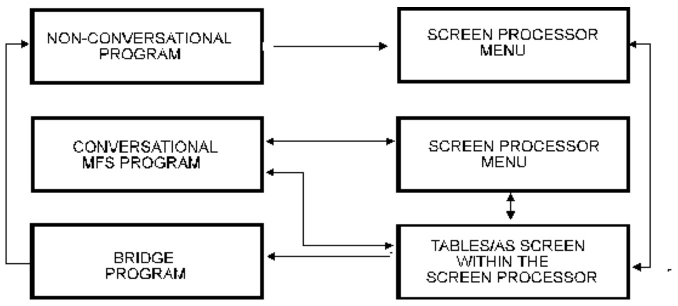
TO SCREEN
Enter a valid **TABLES/AS** screen name in the application being switched to.

OPT CODE
Enter the code or value that will cause the switch to occur. The maximum length is eight characters.

OPT FLD-NAME
Enter the name of the field which will contain the option code value.

See prior information for remainder of fields from transaction code.

Switching To and From Applications Which Use MFS Screens


NON-CONVERSATIONAL PROGRAM
CONVERSATIONAL MFS PROGRAM
BRIDGE PROGRAM
SCREEN PROCESSOR MENU
SCREEN PROCESSOR MENU
TABLES/AS SCREEN WITHIN THE SCREEN PROCESSOR

Switching to and from applications which use MFS screens is similar to switching to and from applications which use **TABLES/AS** screen programs.
Here is a summary of the differences:
• A field number cannot be used for cursor positioning on the target screen.
• Enter a transaction only if the application program is conversational.
• An MFS mod length must be entered.

Types of Screen Switching for MFS Screen Programs

1. From an MFS screen program to the **TABLES/AS** Screen Processor menu.
  A. Where the MFS screen program is non-conversational.
  B. Where the MFS screen program is conversational.

2. From a MFS screen program to a **TABLES/AS** screen within the Screen Processor (

3. From a **TABLES/AS** screen within the Screen Processor to a MFS screen program.
  A. Using a PF key (with or without data transfer; the MFS screen program is either conversational or non-conversational).
  B. Using an option code (with or without data transfer; the MFS screen program is either conversational or non-conversational).

1A. Switching from an MFS Screen Program to TABLES/AS Screen Processor Menu (Non-Conversational)

In order to switch from any non-conversational program with an IMS system to the **TABLES/AS** Screen Processor, the following must be set up. Within the MFS screen, the input field corresponding to the desired program function key defined for switching must contain a minimum of 10 bytes of data in the format:

COLUMN 1-8	Screen processor's transaction code (TS09), or your company's equivalent as defined in the TXNTBL table.
COLUMN 9	Blank character.
COLUMN 10	Character P to indicate that the initiation source is a user program.

When this switch occurs, the Screen Processor menu will appear. A valid screen name will then have to be entered.

NOTE: Non-conversational programs can only switch to the Screen Processor menu. They cannot switch to a **TABLES/AS** screen within the Screen Processor.

There can be NO transfer of data to the Screen Processor menu.

If switching from the Screen Processor back to the application program and the data currently shown on the application screen is to be redisplayed upon return, do the following:

a. In your program, using SQL, insert a record into the **TABLES** control table WORKDB1 using the following instructions.

How to write a screen to the tables intermediate database which is to be restored from the Screen Processor program upon return.

```
01      TABLE-IO-AREA.
        05  IO-AREA-SIZE                          PIC S9(4) COMP VALUE
        05  TABLE-RECORD-KEY.
           10  IO-TERMINAL-ID                     PIC X(8).
           10  IO-SCREEN-NAME                     PIC X(13).
           10  IO-DATA-TYPE                       PIC X(1) VALUE 'B'.
           10  IO-SEGMENT-NO                      PIC S9(4) COMP VALUE
                                                   +1.
        05  IO-SEGMENT-ID                         PIC X VALUE '1'.
        05  IO-SCREEN-DATA                        PIC X(3000).
 WHERE,
        IO-TERMINAL-ID = LOGICAL TERMINAL ID
        IO-SCREEN-NAME = NAME OF THE SCREEN TO BE
            RESTORED, MOD NAME FOR AN MFS SCREEN
        IO-SCREEN-DATA = SCREEN-DATA INCLUDING THE
            LL & ZZ FIELDS FOR AN MFS SCREEN
```

b. A switching record must be defined to switch back to the application program using the Restore option. See items 2, 3A and 3B defined later.

1B. Switching from an MFS Screen Program to Screen Processor Menu (Conversational)

In order to switch from a conversational program within the IMS system to the **TABLES/AS** Screen Processor menu, a SPA must be set up.

A SPA (scratch pad area) size of at least 100 bytes must be communicated to the Screen Processor in the following format:

```
01   SPA
     05      FILLERPIC X(6)
     05      TRAN-CODE                            PIC X(8) VALUE
                          'TS09' OR ITS EQUIVALENT
     05      FILLERPIC X(8)
     05      COND-CODE                            PIC X(1) VALUE ZERO
     05      INITIATION-SOURCE                    PIC X(1) VALUE 'P'
     05      FILLERPIC X(24)
     05      ACCESS-ALLOWED                       PIC X(4) VALUE
                                                     LOW-VALUES.
     05      FILLERPIC X(48).
```

After the SPA is inserted, issue the following call:

```
CALL 'TB2MAINT' USING       CONTROL-AREA
                            SCREEN-COPY-CODE-AREA
                            LOGICAL-TERMINAL-PCB.
```

The SCREEN-COPY-CODE-AREA may be initialized to spaces or a screen name and/or message may be moved in order to display them on the Screen Processor menu.

The record format of the CONTROL-AREA is:

```
01   CONTROL-AREA.
     05    SCREEN-FUNCTION                   PIC X(4).
     05    PFK-PAK-AREA                      PIC 9(2).
     05    CURSOR-POSITION                   PIC S9(4) COMP.
     05    RETURN-CC                         PIC 99.
     05    EDIT-ERROR-FLAG                   PIC X(1).
     05    RECORD-NAME                       PIC X(8).
     05    QUAL-NAME                         PIC X(8).
     05    MAP-OCC-NO                        PIC 9(4) COMP.
     05    SCREEN-PROGRAM-NAME               PIC X(8).
```

The record format of SCREEN-COPY-CODE-AREA for the Screen Processor menu is:

```
01   TB2MAINT
     05    TB2MAINT-TB2SCRN-ATTR-STD         PIC X(1).
     05    TB2MAINT-TB2SCRN-ATTR-EXT         PIC X(1).
     05    TB2MAINT-TB2SCRN-ATTR-COL         PIC X(1).
     05    TB2MAINT-TB2SCRN-ATTR-CUR         PIC X(1).
     05    TB2MAINT-TB2SCRN-DATA                     PIC X(8).
     05    FILLER                            PIC X(36).
     05    TB2MAINT-TB2MESAG-ATTR-STD        PIC X(1).
     05    TB2MAINT-TB2MESAG-ATTR-EXT        PIC X(1).
     05    TB2MAINT-TB2MESAG-ATTR-COL        PIC X(1).
     05    TB2MAINT-TB2MESAG-ATTR-CUR        PIC X(1).
     05    TB2MAINT-TB2MESAG-DATA            PIC X(79).
WHERE
     TB2MAINT-TB2SCRN-DATA = TABLES/AS SCREEN NAME
           TO BE PROCESSED
     TB2MAINT-TB2MESAG-DATA = ANY DISPLAY MESSAGE
```

The record format of LOGICAL-TERMINAL-PCB is:

```
01  LOGICAL-TERMINAL-PCB.
    05    IO-TERMINAL                          PIC X(8).
    05    IO-RESERVE                           PIC X(2).
    05    IO-STATUS                            PIC X(2).
    05    IO-PREFIX                            PIC X(12).
    05    IO-MOD-NAME                          PIC X(8).
    05    IO-USERID                            PIC X(8).
```

If you will be switching from the Screen Processor back to your application program and you want the data currently shown on your application screen to be redisplayed when you return, you must do the following:

a. In your program, using SQL, insert a record into the **TABLES** control table WORKDB1 using the following instructions:

```
    01    TABLE-IO-AREA.
          05  IO-AREA-SIZE                     PIC X9(4) COMP VALUE
                                                      +3027.
          05  TABLE-RECORD-KEY.
                 10  IO-TERMINAL-ID            PIC X(8).
                 10  IO-SCREEN-NAME            PIC X(13).
                 10  IO-DATA-TYPE              PIC X(1) VALUE 'B'.
                 10  IO-SEGMENT-NO             PIC S9(4) COMP
                                                             VALUE +1.
             05  IO-SEGMENT-ID                 PIC X VALUE '1'.
             05  IO-SCREEN-DATA                PIC X(3000).

             WHERE,
             IO-TERMINAL-ID = LOGICAL TERMINAL ID
             IO-SCREEN-NAME = NAME OF THE SCREEN TO BE
                RESTORED, MOD NAME FOR AN MFS SCREEN
             IO-SCREEN-DATA = SCREEN-DATA INCLUDING THE
                LL & ZZ FIELDS FOR AN MFS SCREEN
```

b. A switching record must be defined in **TABLES/AS** to switch back to your screen using the Restore option. See 3A and 3B defined later.

2. Switching from an MFS Screen Program to a TABLES/AS Screen within Screen Processor (Conversational)

Procedure to transfer control to TS09 TXN from another conversational IMS program.

CHANGE MODIFIABLE PCB WITH SCREEN PROCESSOR TRANCODE

```
    MOVE 'TS09'    TO TRAN-TO-TRANSFER.
    MOVE 'TS09'    TO MOD-PCB-LTERM-NAME.
    CALL 'CBLTDLI' USING       IMS-FUNC-CHNG
                               MOD-PCB
                               TRAN-TO-TRANSFER.
    IF MOD-PCB-STATUS = SPACE
            NEXT SENTENCE
    ELSE
            GO TO HANDLE-PROBLEM-STATUS.
```

WHERE, TRAN-TO-TRANSFER IS DEFINED AS FOLLOWS:

```
01 TRAN-TO-TRANSFER                    PIC X(8).
```

NEXT, INSERT SPA TO IMS WITH SCREEN PROCESSOR PCB AND LOW-VALUE

```
    MOVE 'TS09' TO SPA-NEW-TRANCODE.
    MOVE LOW-VALUES TO FILLER-86-BYTES.
    CALL 'CBLTDLI' USING       IMS-FUNC-ISRT
                               MOD-PCB
                               SPA.
    IF MOD-PCB-STATUS = SPACE
            NEXT SENTENCE
    ELSE
        GO TO HANDLE-PROBLEM-STATUS.
```

WHERE, SPA IS DEFINED AS FOLLOWS

```
01  SPA.
    05  FILLER-6-BYTES                 PIC X(6).
    05  SPA-NEW-TRANCODE                       PIC X(8).
    05  FILLER-86-BYTES                PIC X(86).
```

NEXT, INSERT MESSAGE INTO THE IMS MESSAGE QUEUE

```
       MOVE 'SCRNNAME' TO      DEST-IOAREA-SCRNNAME.
       CALL 'CBLTDLI' USING    IMS-FUNC-IRST
                               MOD-PCB
                               DEST-IOAREA.
       IF MOD-PCB-STATUS = SPACE
               NEXT SENTENCE
       ELSE
          GO TO HANDLE-PROBLEM-STATUS.

       WHERE, DEST-IOAREA IS DEFINED AS FOLLOWS:

   01  DEST-IOAREA.
          02  DEST-LL              PIC S9(4)  COMP
                                                   VALUE +20.
          02  DEST-ZZ              PIC X(2).
          02  DEST-IOAREA-SCRNNAME                PIC X(16).

       NEXT, ISSUE A GOBACK.

          GOBACK.

       'SCRNNAME' SCREEN WILL BE AUTOMATICALLY
       EXECUTED THROUGH THE SCREEN PROCESSOR TRANCODE
```

If switching from the Screen Processor back to the application program and the data currently shown on the application screen is to be redisplayed upon return, do the following:

a. In your program, using SQL, insert a record to the **TABLES** control table WORKDB1 using the following instructions:

```
01  TABLE-IO-AREA.
    05  IO-AREA-SIZE                     PIC S9(4) COMP VALUE
                +3027.
    05  TABLE-RECORD-KEY.
       10  IO-TERMINAL-ID                PIC X(8).
       10  IO-SCREEN-NAME                PIC X(13).
       10  IO-DATA-TYPE                  PIC X(1) VALUE 'B'.
       10  IO-SEGMENT-NO                 PIC S9(4) COMP
                  VALUE +1.
    05  IO-SEGMENT-ID                    PIC X VALUE '1'.
    05  IO-SCREEN-DATA                   PIC X(3000).

    WHERE,
          IO-TERMINAL-ID = LOGICAL TERMINAL ID
```

```
IO-AREA-NAME = NAME OF THE SCREEN TO BE
    RESTORED, MOD NAME FOR AN MFS SCREEN
IO-SCREEN-DATA = SCREEN-DATA INCLUDING THE
    LL & ZZ FIELDS FOR AN MFS SCREEN
```

b. A switching record must be defined to switch back to the screen using the Restore option. See 3A and 3B defined later.

3A. Switching from a TABLES/AS screen within Screen Processor to an MFS Screen Program (Using a PF Key)

Obtain the switching screen in **TABLES/AS**, i.e., Opt 7/Opt 3.

Enter the following required fields:

FROM SCREEN
Must be a valid **TABLES/AS** screen.

TO SCREEN
Enter the name of the MFS MOD which is being switched to. To terminate the transaction, enter `*' in the TO screen name.

PF KEY
Enter the PF key number which is to cause the switch.
Valid PF keys are 1-11 and 13-24.

TRANSACTION CODE
- If the MFS screen is conversational, enter the transaction code of the conversational program being switched to.
- If the MFS screen is non-conversational, leave the transaction code field blank.

MFS MOD LENGTH
Enter the MOD length of the MFS screen being switched to.

OPTIONAL FIELD

EDITOR PROGRAM NAME
To call a user-written program before the TO screen is displayed, enter the editor program name.

MAP NAME TO TRANSFER
Enter the table, map or IMS segment name of the record being transferred.

The Screen Processor will send the screen and transfer the following SPA to the application program if it is conversational.

```
01   SPA
     05     FILLERPIC X(6)
     05     TRAN-CODE                             PIC X(8) THE
      TRANSACTION CODE OF THE APPLICATION PROGRAM
     05     SCREEN-ID                             PIC X(8) THE NAME OF
                 THE SOURCE SCREEN
     05     CONDITION-CODE                        PIC X(1)
     05     INITIATION-SOURCE                     PIC X(1)
     05     FILLER                                PIC X(76) VALUE
                 LOW-VALUES
```

SAVE AND RESTORE DATA

- Enter S to save data fields on the FROM screen before switching.
- Enter an R to restore data fields previously saved for the TO screen when switching occurs.
- Enter a B to both save and restore.
- Leave the field blank if saving or restoring is not desired.

3B. Switching from a TABLES/AS Screen within Screen Processor to an MFS Screen Program (Using an Option Code)

Obtain the switching screen in **TABLES/AS**, i.e., Opt 7/Opt 3.

The rules for retrieving, updating, and transferring data are identical to PF key option.

The following are the required fields.

FROM SCREEN
Must be a valid **TABLES/AS** screen name.

TO SCREEN
Enter the name of the MFS mod which is being switched to.

OPT CODE
Enter the code or value that will cause the switch to occur. The maximum length of this value is eight characters.

OPT FLD-NAME
Enter the name of the field which will contain the option code value.

See previous information for remainder of fields from Transaction Code.

4. Switching from a TABLES/AS Screen within Screen Processor to Non-AS Transactions.

To allow switching from an IMS conversational transaction (e.g., **TABLES/AS**) to a non-conversational transaction (e.g., user program) using a program-to-program switch requires a special bridge program. This allows the conversational transaction to directly switch to the non-conversational one without the user doing something on the terminal.

The bridge program acts as a conversational transaction that handles the conversational aspects (e.g., GU/ISRT to the SPA) and dynamically calls the non-conversational program to do its normal work. The non-conversational program, with very minor changes, can run normally, that is, started by user input or called from the bridge program. A sample bridge program is shipped with **TABLES** for your use. It resides in SOURCE library as member CNBRIDGE.

To allow the non-conversational program to work as a standard transaction or to be called from the bridge program the following change must be made:

Where the non-conversational program does a GU to the message queue, it must do either a GU or a GN depending on how it is called. If the program currently does the following:

```
CALL 'CBLTDLI' USING          IMS-FUNC-GU,
                              LOGICAL-TERMINAL-PCB,
                              MESSAGE-AREA.
```

It can be changed to do the following:

```
IF LT-PCB-STATUS = '99'
    CALL 'CBLTDLI' USING      IMS-FUNC-GN,
                              LOGICAL-TERMINAL-PCB,
                              MESSAGE-AREA
IF LT-PCB-STATUS = 'GC' OR 'GD'
    MOVE SPACES TO LT-PCB-STATUS
    MOVE SPACES TO MESSAGE-AREA
    END-IF
    . . . . ANY OTHER ADDITIONAL LOGIC
ELSE
    CALL 'CBLTDLI' USING      IMS-FUNC-GU,
                              LOGICAL-TERMINAL-PCB,
                              MESSAGE-AREA
    END-IF.
```

The LT-PCB-STATUS field is in the LOGICAL-TERMINAL-PCB. The value '99' is not a valid IMS status but is actually set in the bridge program. When the bridge program calls the non-conversational program it will do a GN and allow a

'GC' or 'GD' status to be valid. When the transaction is called normally, it will do the 'GU' as normal.

The non-conversational program can then perform the logic it was coded to perform.

Notes: This change will not affect the normal operation of the transaction. It will still function as a normal non-conversational transaction.

This change is based on using the sample bridge program. function.

See the last section on What Is Passed below for possible other changes that might be required.

BRIDGE PROGRAM SETUP REQUIREMENTS

1. Using the sample bridge program, create a program that calls the non-conversational program with the correct PCB's. Compile and link it.

Note: The call to the non-conversational program must be set up to pass the correct PCB's that are expected.

---

2. Do an IMS gen to define a new conversational transaction that points to the bridge program just created and a PSB that contains the same PCB's as the non-conversational program.

3. Define security for the new conversational transaction exactly the same as for the non-conversational one, if required.

4. Change the non-conversational program as described above. This can be done anytime as the change will not affect the normal operation of the program.

TABLES\AS SWITCHING

In **TABLES/AS**, define a screen switch to the new transaction defined above. When it gets initiated, the bridge program simply calls the non-conversational program to display a message or MFS screen. The non-conversational transaction is then in control.

Obtain the switching screen in **TABLES/AS**, i.e., Option 7/Option 3.

Enter the following fields:

FROM SCREEN
Enter the AS screen name you are switching from.

TO SCREEN
Enter '*' (i.e., exiting from **TABLES/AS**).

PF KEY
Enter the PF key number which is to cause the switch.
Valid PF keys are 1-11 and 13-24.
(Optionally use an option field and value.)

MFS MOD LENGTH
Leave blank (denotes no MFS).

TRANSACTION CODE
Enter the transaction code for the conversational bridge program created above.

WHAT IS PASSED

When **TABLES/AS** switches to another transaction, it does a change to the modifiable PCB for the new tran code and inserts a message. The message is made up of the length field (e.g., LLZZ) and 8 bytes for the transaction code followed by 1 space. So, if switching to a bridge transaction called BRIDGE1 that actually calls the non-conversational program, NONCONV1, then when NONCONV1 does the GN to the message queue, it will receive the following:

```
'llzzBRIDGE1  '
```

in the input message area. The llzz field will contain the hex value x'000D0000' (length of 13 = 4 for LLZZ, 8 for the trancode, and 1 for a space). BRIDGE1 is the name of the trancode and it will be followed by 2 spaces.

IMPORTANT
If the non-conversational program expects a different tran code or other information in the first message, some additional changes may be required to the non-conversational program. This can also be checked and handled when doing the changes as described above.

Appendix H
Screen Switching (CICS)

Screen switching involves transferring from one screen to another screen. This transfer may involve:

- Transferring data from one screen to the other
- Saving the data on the screen being transferred FROM
- Restoring data on the screen being transferred TO. The data must have been saved previously.

The following write-ups cover the capabilities and coding requirements for:

- Switching TO and FROM applications which use **TABLES/AS** screens.
- Switching TO and FROM applications which use BMS screens.

The options provided for switching to/from **TABLES/AS** within a CICS environment are:

- Switch to the Screen Processor menu
- Switch to a screen within the Screen Processor
- Save your screen data for redisplay upon return from a screen within the Screen Processor
- Pass **TABLES/AS** required Commarea of 100 bytes
- Pass additional data in the Commarea up to 32,000 bytes
- Return to your program the **TABLES/AS** Commarea of 100 bytes
  or
  Return the additional data in the Commarea - Note a max of 8000 bytes are returned
- Pass data to **TABLES/AS** to be used for searching

SAVE SCREEN DATA FOR REDISPLAY

a. In your program, write a record, using SQL insert, into the **TABLES/AS** control table WORKDB1 using the following instructions:

```
01 TABLE-IO-AREA.
    05 IO-AREA-SIZE              PIC S9(4)
                                           COMP VALUE +3027.
    05 TABLE-RECORD-KEY.
        10 IO-TERMINAL-ID        PIC X(8).
        10 IO-SCREEN-NAME        PIC X(13).
        10 IO-DATA-TYPE          PIC X VALUE 'B'.
        10 IO-SEGMENT-NO         PIC X9(4)
                                           COMP VALUE +1.
    05 IO-SEGMENT-ID             PIC X  VALUE '1'.
    05 IO-SCREEN-DATA                PIC X(3000).
WHERE,
  IO-TERMINAL-ID   =  LOGICAL TERMINAL ID
  IO-SCREEN-NAME   =  Name of the screen to be restored, the Map
                         name for a BMS screen.
  IO-SCREEN-DATA   =  Screen Data
```

b. Using the **TABLES/AS** Switching Definition Panel, define a switch back to your application program using the Restore Option. See 'Returning to Your Program' defined later.

COMMAREA FOR INTERFACE TO/FROM TABLES/AS

```
01 TABLESAS-COMMAREA.
    05 TABLESAS-DATA.
        10 USERS-DATA           PIC X(6).
        10 TRAN-CODE            PIC X(8).
        10 SCREEN-IDENT         PIC X(8).
        10 COND-CODE            PIC X(1).
        10 INITIATION-SOURCE PIC X(1).
        10 FILLER               PIC X(69).
        10 KEY-STROKE           PIC 9(02).
        10 FILLER               PIC X(5).
    05 APPL-DATA             PIC X(1000).

* Max APPL-DATA is 32000.  Only first 8000 are returned
* when switching is requested.
```

TRANSFER TO TABLES/AS PASSING A COMMAREA WITHOUT SEARCH FIELDS

Assume that a COMMAREA of 1000 bytes called APPL-DATA is to be transferred to **TABLES/AS** but no data is to be passed to perform automatic retrieval before displaying the screen.

In addition to the Commarea, the following coding is required:

```
*
* TO TRANSFER CONTROL TO TABLES/AS, USE:
*
        MOVE LOW-VALUES   TO TABLESAS-DATA.
        MOVE 'scr-name'  TO SCREEN-IDENT.
        MOVE 'X'          TO COND-CODE.
        MOVE 'P'          TO INITIATION-SOURCE.
        EXEC CICS XCTL
                  PROGRAM('TSCM0900')
                  COMMAREA(TABLESAS-COMMAREA)
                  LENGTH(1100)
        END-EXEC.

* Notes:
*     -   Must have a screen name specified to be displayed.
*     -   Must have an entry in the TXNMNU control table that has the 'scr-name'
*          moved to SCREEN-IDENT
*          & a 'transaction code' to run that 'scr-name' under.
*     -   Must have an entry in the PCT table for the 'transaction code' specified in
*          the TXNMNU table.  The PCT entry must point to program TSCM0900.
*     -   Must have an entry in the RCT table pointing to a DB2 plan.
```

TRANSFER TO TABLES/AS PASSING SEARCH DATA AND A COMMAREA

Assume that a COMMAREA of 1000 bytes called APPL-DATA is to be transferred to **TABLES/AS** and data is to be passed to perform automatic retrieval before displaying the screen

In addition to the Commarea, the following coding is required:
01 'scr-name' Copy Code Member.

```
*
* TO TRANSFER CONTROL TO TABLES/AS, USE:
*
        MOVE SPACES                TO 'scr-name'-copy-code-area.
        MOVE 'search value1'       TO 'scr-name'-copy-code-field1
        MOVE 'search value2'       TO 'scr-name'-copy-code-field2
        MOVE 'search valueX'       TO 'scr-name'-copy-code-fieldX
        MOVE LOW-VALUES            TO TABLESAS-DATA.
        MOVE 'scr-name'            TO SCREEN-IDENT.
        MOVE 'Y'                   TO COND-CODE.
        MOVE 'P'                   TO INITIATION-SOURCE.
        MOVE 'keystroke'           TO KEY-STROKE.
        EXEC    CICS RETURN
                TRANSID('tran-code')
                COMMAREA(TABLESAS-COMMAREA)
                LENGTH(1100)
                IMMEDIATE
                INPUTMSG('scr-name' Copy-Code-Area)
                INPUTMSGLEN('scr-name' Copy-Code-Area length)
              END-EXEC.

*
* Notes:
*    -  This capability requires a CICS level that supports the use of IMMEDIATE.
*    -  Set the COND-CODE value to 'Y'.
*    -  Must have a screen name specified to be displayed.
*    -  Must have an entry in the TXNMNU control table containing:
            The 'tran-code' specified by TRANSID
           & The 'scr-name' moved to SCREEN-IDENT.
*    -  Must have an entry in the PCT table for the given
*       'tran-code' pointing to TABLES/AS program TSCM0900.
*    -  Must have an entry in the RCT table pointing to a DB2 plan.
```

TO RETURN CONTROL BACK TO APPLICATION TRANSACTION

a.* Define a switch using **TABLES/AS** Switching Definition Panel:
* This example will return the APPL-DATA portion of the
* Commarea

Enter FROM Screen name,
Enter '*' in the TO Screen name,
Enter a PF Key or Field value to invoke switching
Enter a Transaction code to return control to
Enter a literal 'COMMSAVE' in the Map name field

* Note:
*` 'COMMSAVE' is the keyword in this example.
* Only the APPL-DATA portion of the Commarea, up to 8000
* bytes, will be returned to the application transaction.
* The first 100 bytes, TABLESAS-DATA, is not returned.

b. * Define a switch using **TABLES/AS** Switching Definition Panel:
* This example will return the TABLESAS-DATA portion of the Commarea

Enter FROM Screen name,
Enter '*' in the TO Screen name,
Enter a PF Key or Field value to invoke switching
Enter a Transaction code to return control to
Enter a literal 'COMMAREA' in the Map name field

* Note:
* 'COMMAREA' is the keyword in this example.
* Only the 1st 100 bytes of the Commarea are returned.
* The 1st 6 bytes of the Commarea are unchanged. The
* TRAN-CODE and SCREEN-IDENT will contain the values
* associated with the specific screen used when the switch
* from **TABLES/AS** was invoked.

c.* Define a switch using **TABLES/AS** Switching Definition Panel
* This example will start the given CICS transaction at interval 380
* without returning any Commarea

Enter FROM Screen name,
Enter '*' in the TO Screen name,
Enter a PF Key or Field value to invoke switching
Enter a Transaction code to return control to

d. * Define a switch using **TABLES/AS** Switching Definition Panel:
* This example will send the given BMS map and either return
* to CICS or return to CICS with the given transaction id.

Enter FROM Screen name,

Enter TO Screen name, BMS map name
Enter a PF key or Field value to invoke switching
Enter a Transaction code to return control to or leave it blank,
Enter an 'R' in the Save/Restore option to restore the BMS map previously saved

Switching from a TABLES/AS Screen within Screen Processor (Using a PF Key)

Obtain the switching screen.

Enter the following required fields:

FROM SCREEN
Enter a valid **TABLES/AS** screen name.

TO SCREEN
Enter the name of the BMS map which is being switched to. Enter the name of the **TABLES/AS** screen in the application being switched to. To terminate the transaction, enter `*' in the TO screen name.

PF KEY
Enter the PF key number which is the cause the switch.
Valid PF keys are 1-11 and 13-23.

TARGET MAP LENGTH
Enter the MAP length of the BMS screen being switched to.
To terminate the transaction, enter 040 in the Map Length.

TRANSACTION CODE
Enter the transaction code of the program you are switching to.
To terminate the transaction code, leave this field blank.

OPTIONAL FIELDS

EDITOR PROGRAM NAME
To call a user-written program before the TO screen is displayed, enter the editor program name.

TRANSFERRING DATA
To transfer data from the FROM screen to the TO screen:

MAP NAME TO TRANSFER
Enter the map name of the record being transferred.
To transfer data from the **TABLES/AS** screen to a BMS screen, first create a record map from the FROM screen to the BMS copy code. Enter that record map name here.

The fields on the screen defined in the record map will be displayed on the BMS screen when the switch occurs.

If data is not to be passed from one screen to the other, leave this field blank.

SAVE AND RESTORE DATA
- Enter an S to save data fields on the FROM screen before switching.
- Enter an R to restore data fields previously saved for the TO screen when switching occurs.
- Enter a B to save and restore.
  - Leave the field blank if saving or restoring is not desired.

Switching from a TABLES/AS Screen within Screen Processor (Using an Option Code)

Obtain the switching screen.

The rules for retrieving, updating, and transferring data are identical to PF Key option.

The following are the required fields:

FROM SCREEN
Enter a valid **TABLES/AS** screen name. This screen name will have to be entered on the Screen Processor menu when the switch is to take place.

TO SCREEN
Enter the name of the BMS map which is being switched to or a **TABLES/AS** screen in the application.

OPT CODE
Enter the code or value that will cause the switch to occur. The maximum length is eight characters.

OPT FLD-NAME
Enter the name of the field which will contain the option code value.

Appendix I
TABLES/AS Abends

In general, **TABLES/AS** will issue an abend only for situations which should generally not occur and that cannot be handled by issuing a message. In most abnormal situations, a message is displayed indicating the problem. For situations that cannot be handled, the following abends may occur:

2000	Generic abend code issued by online transactions when an unknown situation occurs.
2001	An insert to the SPA has failed (IMS **only**).
2002	A CICS SEND or IMS ISRT to DFSM04 returned a bad return code trying to send a message to the terminal.
2003	A CICS SEND or IMS ISRT to DFS.EDTN returned a bad return code trying to send a full screen panel to the terminal
2004	A CICS SYNCPOINT ROLLBACK failed. The abend is issued to insure ROLLBACK does occur (CICS **only)**.

NOTE: In all cases, the dump should show the IMS STATUS CODE or CICS EIB STATUS. If the problem cannot be solved by checking the status in the dump, save the dump and call SSI Technical Support.

Appendix J
Disk Space Estimating

Specialized Software DB2 Control Tables

'AVERAGE SCREEN REQUIREMENTS'

1 DB2 table processed per screen - Add/Delete/Change/Inquire Functions

Each table has Effectivity Control

Each table has 5 columns

All columns have Field Edits defined

1 Cross Table Validation required per screen

Customize message for all edits

1 Common Message field

A Function Code field

1 Control field for Relational Edits - 3 Conditions to be defined

Each screen will be locked

Log all activity - A/D/C - 75% to be Changes

Multiple tables will NOT be processed on a screen

See sample screen format at end of this appendix.

Control Table	**Avg Bytes / Row**	**Parameters**
APPLID	68	(1/Unique Set of Transactions) assigned to access TABLES - Normally < 20 rows
AS_MESSAGES	VAR-808 MIN-0<264	Depends on message length.
AUDITDB	VAR-4027 Min-200	Depends on: (DB2 logging set to Y Size of table being logged Activity A/D-1 row C-2rows Frequency of archive)
IMSSEG	346	(1/VSAM file or /IMS-DB SEG) to be accessed by TABLES
IMSSE2	44	(1/IMS-DB SEG) to be accessed by TABLES
LAYOUT	182	(1/Copycode Field/VSAM File or /IMS-DB SEG) to be accessed by TABLES
MS_DEFINITION	558	(2/Table Name with Effectivity Control) + (2/Table Name Accessed by Memory Manager with 'Order By' defined)
SCPMSG	92	(1/Numbered Processing Message overridden /Screen) - not usually overridden
SCRCHK	91	(1/Line/Condition/Relational Control Field)
SCRCON	143	(1/Field/Screen)
SCRDFT	94	(1/Table Processed/Screen)
SCREDT	16	(1/Screen) **IF** an Editor will control all screen processing
SCRERR	97	(1/Customized Message/Field/Screen)
SCRINT	72	(1/Field/Screen) for each field with Initial Values defined. All **AP** & **AS** Default screens have Date & Time initialized
SCRLOK	34	(1/Screen)

Control Table	**Avg Bytes**	**Parameter**

	/ Row	
SCRMSG	19	(1/Common Message Field/Screen)
SCRORD	88	(1/Field/Map/Screen with Sort NO & S specified
SCRRCD	217	(1/Table Processed/Screen) - Allow Functions
SCRSWC	98	(1/Screen Switch) - Utilities - Switching
SCRVAL	235	(1/Field/Screen) for each field with edits Each field Mapped to a DB2 table will have edits defined
TXNMNU	16	(1/User Transaction Code) assigned to initiate a TABLES/AS screen to be displayed
TXNTBL	130	(1/APPLID)
USERID	37	(1/User/VSAM File or /IMS-DB SEG) that the user is authorized to access via TABLES
VALDTE	45	(1/Line of Cross Table Validation Defined/Map/Screen)
VSMSE2	542	(1/VSAM File) to be accessed via TABLES
SCREENDB	Max-4012	Maximum of 8 rows per screen. Row sizes vary considerably from screen to screen. Row sizes vary considerably within a screen. Maximum is 32,000 **AVG SCREEN** is 6,000 Row S = 3800 Row 1 = 700 Row R = 400 Row N = 1100
WORKDB1	MAX-4027	Varies by number of users, by number of rows retrieved that must be stored for display or update, etc...this is temporary work area
WORKDB2	MAX-4029	Varies by number of users, by number of rows retrieved that must be stored for display or update, etc...this is temporary work area

SUGGESTED SPACE PER 100 AVG SCREENS

	APPLID	1,400	
*	AUDITDB	1,000,000	
	AS_MESSAGES	30,000	
	IMSSEG	7,000	
	IMSSE2	4,000	(IMS)
	LAYOUT	18,500	
	MS_DEFINITION	111,600	
	SCPMSG	4,600	
	SCRCHK	81,900	
	SCRCON	128,700	
	SCRDFT	9,400	
	SCREDT	-0-	
	SCRERR	145,500	
	SCRINT	28,800	
	SCRLOK	3,400	
	SCRMSG	1,900	
	SCRORD	17,600	
	SCRRCD	21,700	
	SCRSWC	29,400	
	SCRVAL	117,500	
	TXNMNU	1,600	
	TXNTBL	2,600	
	USERID	7,400	
	VALDTE	4,500	
	VSMSE2	10,840	(CICS)
*	SCREENDB	600,000	
*	WORKDB1	1,000,000	
*	WORKDB2	1,000,000	

Those tables with an * have the most chance for volatility and should be monitored most closely.

In addition to the number of bytes shown, a 25% safety factor should be considered.

TABLESPACE NAME	CONTROL TABLE
SSIPAU41	AUDITDB
SSIPSC41	SCREENDB
SSIPWK41	WORKDB1 & WORKDB2
SSIPCT41	ALL OTHER SSI CONTROL TABLES

```
TABLE NAME:  SSI006.CARCL41---------------------------------------DATE: AAAAAAAA
               SCR206A SCREEN in AUTHID - SSI007                   TIME: AAAAAAAA
              A RATE OPERATOR
F CAR-CLASS MILEAGE-LIMIT DAILY-RATE EFFECTIVE-DATE EXPIRATION-DATE
- --------  ------------- ---------- -------------- ---------------
A A       QQQQ        QQQQQQQ   DDDDDDDDDD    DDDDDDDDDD
A A       QQQQ        QQQQQQQ   DDDDDDDDDD    DDDDDDDDDD
A A       QQQQ        QQQQQQQ   DDDDDDDDDD    DDDDDDDDDD
A A       QQQQ        QQQQQQQ   DDDDDDDDDD    DDDDDDDDDD
A A       QQQQ        QQQQQQQ   DDDDDDDDDD    DDDDDDDDDD
A A       QQQQ        QQQQQQQ   DDDDDDDDDD    DDDDDDDDDD
A A       QQQQ        QQQQQQQ   DDDDDDDDDD    DDDDDDDDDD
A A       QQQQ        QQQQQQQ   DDDDDDDDDD    DDDDDDDDDD
A A       QQQQ        QQQQQQQ   DDDDDDDDDD    DDDDDDDDDD
A A       QQQQ        QQQQQQQ   DDDDDDDDDD    DDDDDDDDDD
A A       QQQQ        QQQQQQQ   DDDDDDDDDD    DDDDDDDDDD
A A       QQQQ        QQQQQQQ   DDDDDDDDDD    DDDDDDDDDD
A A       QQQQ        QQQQQQQ   DDDDDDDDDD    DDDDDDDDDD
A A       QQQQ        QQQQQQQ   DDDDDDDDDD    DDDDDDDDDD
A A       QQQQ        QQQQQQQ   DDDDDDDDDD    DDDDDDDDDD

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAA
PF1-HELP PF2-MAKES PF3-CLASS PF4-MENU  PF11-TP MONITOR   PF12-EXIT
```

Appendix K
Migrating to Production - Considerations

When migrating from one environment to another, the following items need to be considered:

- Are the transaction codes the same in both environments?
  - Migrate the custom transaction code
  - Check switching records if transaction code was not '*'.

- Are screen names the same in both environments?
  - Check the TO SCREEN names in Switching records.
  - Check the TO SCREEN names in Process Options - Allow Functions.

- Are table names the same in both environments?
  - Check JCLMIG02 to/from table names (Steps A & C)
  - Insure static SQL module has correct table name.

- Are the plan names the same in both environments?

- Are static SQL modules to be used?
  - Are fully qualified table names used in the module?
  - Are static SQL names the same in both environments?
    - Check Process Options - Allow Functions

- Are screen load modules to be used?

- Are there editor programs involved?
  - Are editor program names the same in both environments?
    - Check Switching Definitions?
    - Check Process Options - Allow Functions

- Have effectivity definitions been added or changed?

- Do we lock **TABLES/AS** screens in production?

- What security is needed for the transaction?

- What security is needed for the tables?

- Was JCLMIG02 run while the target TP monitor was active?
  - SSI control tables may need to be reloaded.

Migrating to Production

MIGRATE CUSTOMIZED TABLES/AS TRANSACTION CODE

This process is done for each new transaction code being installed.

A. Each new **TABLES/AS** transaction that is invoked by a user must be entered in the **TABLES/AS** control table **TXNMNU** using the TSAP transaction. An example of a user-invoked transaction would be a transaction code entered by a user to display a menu screen.

B. In CICS, each new TS09 equivalent transaction must be included in the PCT pointing to TSCM0900 program.

C. In IMS, each new TS09 equivalent transaction must be included in the IMS GEN.

   The program name may be TSIM0900 or another program name may be specified. If another program name is specified, it should be an alias name for program TSIM0900.

   If an alias name was used, a PSB GEN must be run for that name. A renamed copy of the TSIM0900 PSB may be used.

ESTABLISH DB2 PLAN

Each new transaction should point to a new or existing DB2 plan.

A. Each DB2 plan must be bound and authorized.
   - Keep in mind that static SQL modules product DBRMs. All SQL modules that are executed from a given plan must be bound together.
   - Each plan must include all DBRM members supplied by SSI for its plan TSCM0900 or TSIM0900.

B. Each new transaction and its corresponding plan must be included in the DSN RCT table in CICS.

CREATE STATIC SQL MODULES IF DESIRED

A static SQL module may be created for each table processed by a screen via JCLSQL01.

A. Move the static SQL source into a production library.

B. If static SQL was created with a fully qualified DB2 table name, change the fully qualified test DB2 table name to the fully qualified production DB2 table name.

C. Compile static COBOL module into a production load library.
   - You must include the resulting DBRM member in your DB2 plan.

D. Add an entry in the production PPT for the static COBOL program name in CICS.

CREATE A SCREEN LOAD MODULE IF DESIRED

A screen load module may be created for any screen.

A. If you are using a different naming standard for your production screens, consider the following:
   - You must copy your test screen under the production name on your test system itself.
   - You may want to lock your production screen using the **TABLES/AS** control table SCRLOK. This can be done using the TSAP transaction on your test system. Once a screen is locked, it cannot be changed without unlocking it.

B. Using JCL401M7, create a static screen module.

C. Move the static screen load module into your production library.

D. Add an entry in the production PPT to include static ASSEMBLER screen program (CICS only).

E. Migrate the screen format and related information.
   - Extract the screen using STA of JCLMIG02. If table name change is required, specify the corresponding production table name under the DD name CHGDB2TB.
   - Keep in mind that a PDS file may be specified for CHGDB2TB which may contain all test and production table names for all screens.
   - The user running the extract job must be authorized to perform the SELECT access to all **TABLES/AS** control tables. This is required because the extract program uses dynamic SQL calls.
   - Load the screen extracted in STA above into the production system. The load should be performed using STB of JCLMIG02.

- The load step uses a DB2 plan which should be granted for EXECUTE to the user running the load job. The load program uses static SQL calls only and therefore GRANT SELECT is not required.

MIGRATE EFFECTIVITY DEFINITIONS

This is required when a DB2 table has been set up with effectivity controls. It is usually migrated only once.

A. Extract effectivity definition from the test system using STC of JCLMIG02. If a table name change is required, specify the corresponding production table name under the DD name CHGDB2TB.

Keep in mind that a PDS file may be specified for CHGDB2TB which may contain all test and production table names for all screens. This PDS file may be the same file which was used during the STA of screen migration.

The extract step uses a DB2 plan which should be granted for EXECUTE to the user running the extract job. The extract program uses static SQL calls only and therefore GRANT SELECT is not required.

B. Load effectivity definition using STD of JCLMIG02.

The load step uses a DB2 plan which should be granted for EXECUTE to the user running the load job. The load program uses static SQL calls only and therefore GRANT SELECT is not required.

MOVE USER EDITOR PROGRAMS

You must migrate the user-written editor programs into your production libraries.

A. Move source and load modules. No recompilation is required.

B. Add entries in the production PPT to include COBOL editor programs (CICS only).

ESTABLISH SECURITY

You must use your standard security procedures to secure transactions and tables.

A. Transactions should be secured using RACF, ACF2, etc.

B. DB2 tables should be secured by DB2 plans.

NAMING REQUIREMENTS

The names used for the static SQL modules, static screen modules and the user-written editor programs must all be different.

TABLES/AS CONTROL TABLE REFRESH

If migration is performed while the production IMS/CICS system is running, then you will need to refresh the **TABLES/AS** control tables using one of the methods below:

- Recycle CICS/IMS
- Run TSLU transaction to free **TABLES/AS** control tables
- Run TSO1 transaction and execute the Utility Option 7 to reload **TABLES/AS** control tables.

Custom Transaction Code - Considerations

Assign transaction code to TSCM0900 / TSIM0900 programs.

GEN. PCT in CICS or perform IMS GEN.

Add TXN.NAME / screen name in TXNMNU control table using **TABLES/AP** processing screen.

IF there is any switching defined, insure that the switching records refer to the new transaction code in both the TO and FROM switching records
or
The switching records all have * for transaction code.

Add DB2 plan name in CICS RCT table for new TXN.

NOTE:
1. All PF keys from screen may be defined (00, 01-24)
2. Standard PF keys 12 and 24 reserved by screen processor are disabled and they can be defined by user.

WARNING: An EXIT must be defined from a screen or group of screens initiated by a custom transaction code since PF12/24 are **NOT** reserved for the screen processor when a custom transaction code is used.

DB2 Column Sub-Defined

There are times, especially when performing a straight conversion of a legacy system to a DB2-based system, when a DB2 column does not contain normalized data. In reality, the DB2 column is comprised of several fields.

TABLES/AS provides the ability to merge imported copy code along with information from the DB2 catalog, thus allowing for the design of screens where the fields of the DB2 column may be used as independent fields on the screen; i.e., **TABLES/AS** will break up the DB2 column before display and will rebuild it before performing an add or update.

To utilize this capability, the following steps could be followed:

1. Run DCLGEN for the DB2 table.

2. Create a copy code member of the COBOL declaration.

3. Insure copy code field names for each of the fields which do NOT have to be sub-defined are DIFFERENT than the DB2 column name. Note that there is a difference between underscore (_) and dash (-).

4. For each of the DB2 columns to be sub-defined, make certain the copy code name is the SAME as the DB2 name. The only changes needed are to change the dashes in the column name to underscores.

5. Create level 15 subdefinitions under the columns required.
   - Remove the PIC clause following the DB2 column name and end the name with a period.
   - Insert level 15 breakdowns that account for the full length of the DB2 column.

6. Run JCL501MC to bring the copy code into **TABLES/AS**.
   - The copy code member name must be the name of the member just modified.
   - The name used in the CONTROL DD may be the same as the copy code member name. This name is the name that **TABLES/AS** will require when mapping to the screen.

7. Design your screen as if the DB2 column is actually in the format of the sub-defined fields.

8. During the MAPPING process, specify the source as DB2 and enter the fully qualified DB2 table name. Also enter the 'copy code member name'. This is the name specified in the CONTROL DD when JCL401MC was run.

When utilizing a screen with a sub-defined field of a DB2 column:

- All of the sub-defined fields should be on the screen.
- Retrievals based on data in the DB2 column will use all of the sub-defined fields and will only allow for an equal condition. Leaving all sub-defined fields empty will perform an unqualified retrieval.
- Search operators have no effect when used with sub-defined DB2 columns.

Optional Components

Component - A

TABLES/AS Batch Update

Overview

TABLES/AS Batch Update provides a facility to process transactions from a batch input file, one record at a time, utilizing edit definitions pertinent to a particular DB2 table as defined for a **TABLES/AS** screen. Inserts, changes and deletions may be processed independently or co-mingled in a single run. Only update records which pass all edits defined are applied.

This facility insures that maintenance performed in batch will process through the same edits as activity processed on-line through the **TABLES/AS** screen.

Three sets of JCL are supplied:

JCLUPD01	- Generalized Batch Update
JCLMASSC	- Mass Change Process
JCLMASSD	- Mass Delete Process

JCLUPD01 - Generalized Batch Update

This utility uses DB2 plan TBLASBCH.

Generalized Batch Update reads transactions from a batch input file one record at a time and processes them using the definitions stored for a **TABLES/AS** screen. Inserts, deletes and changes can be applied to the DB2 table(s) defined for the particular screen.

If the screen name entered is '*DEFAULT,' then **TABLES/AS** Batch builds and stores a **TABLES/AS** on-line default screen dynamically, and processes the input using the default screen.

When a customized **TABLES/AS** screen is specified, editing and validations defined for processing the screen on-line are applied to the batch input. Also, effectivity processing, relational editing and the multi-table updating functions of **TABLES/AS** are all performed without programming. If an editor program or a static module is used to perform specific processing for the on-line screen, the modules must be regenerated using the JCL supplied with **TABLES/AS** batch product, and must be compiled and linked using the DB2 TSO attach interface. Also, the editor and static SQL modules must be kept in the **TABLES/AS** batch load library or a library other than the one used for the on-line system so that the on-line system can continue to execute using its own modules.

Resulting errors are reported in a batch printout at the end of the batch execution.

A report identifying columns which failed the edits is produced. Rows with edit errors are not added to the DB2 table. The report prints the row in character and hex and lists which columns did not pass the edits. A count of records loaded and failed is provided.

The records in the update file will be applied in the sequence they are read from the update file. The only restriction on sequencing is that when change transactions involve an Old/New record, they must follow each other and throughout the entire update file, all change transactions must be in the sequence specified in the TRAN control card, columns 50-52.

Batch Updating provides the ability to take a sequential data set in the DCLGEN format for the table and insert the rows into the DB2 table. This is the simplest way to use the batch updating facility. It can be accomplished by specifying '*DEFAULT' for a screen name, by using a **TABLES/AS** screen created with default = YES or with custom **TABLES/AS** screens that contain all the table fields mapped in the same sequence as the DB2 DCLGEN output.

For screens created with default = SEL or where fields have been removed from the screen, the update file must contain only the fields in the map and in the DCLGEN format for each column.

TABLE FIELDS	MAPPED SEQUENCE	COLUMN SEQ IN UPDATES
FLD1	FLD3	FLD1
FLD2	FLD2	FLD2
FLD3	FLD1	FLD3
FLD4	FLD4	FLD4
FLD5	FLD6	FLD6
FLD6		
FLD7		

In the update records, no space should be provided for FLD5 and FLD7. The column format in the update records should be the format provided by a DCLGEN.

To see the exact DCLGEN format required for customized screens, (non *DEFAULT), you should run JCLSQL01 supplied with the product and review the output.

You may specify a previously defined **TABLES/AS** screen to perform editing on the data being loaded. All of the field and relational editing capabilities of **TABLES/AS** can be used.

The screen must have been 'PREPARED' in **TABLES/AS** in order to utilize the defined edits. When screen name = '*DEFAULT' no prepare is necessary.

You may specify that all existing rows in the table are to be deleted prior to processing rows from the extract file.

Description of the Batch Control Card Input

TABL - Table control record - TABL

This card specifies a fully qualified DB2 table name and optionally, a screen name which was designed to update the given table. If a screen is specified, its definition must exist in **TABLES/AS**. If it has multi-table update capability defined, the table name specified in this control record must be the name of the first table processed. If there are multiple 'processing maps' defined for this table, a map name should be entered to insure that it is used.

Columns 01-04
- 'TABL' Constant to identify this card.

Columns 10-17
- **TABLES/AS** screen name for edits.
- '*DEFAULT' - To create a **TABLES/AS** default screen.

Columns 19-26
- **TABLES/AS** map name used in this screen to qualify the specific map to use to process this table.

Columns 28-63
- Fully qualified DB2 table name or
- Fully qualified DB2 view name or
- Fully qualified DB2 alias name or
- Fully qualified DB2 synonym name

Alias name and synonym name are only valid if the screen name entered is '*DEFAULT'.

TRAN - Transaction control record - TRAN

This card is optional. It specifies what update functions are to be performed on the given table.

Columns 01-04
- 'TRAN' Constant to identify this card.

Columns 10
- 'Y' to delete all table data before starting update
  else . . leave blank

Columns 12-15
- The start position within the update records where the transaction code will be found. First position in the update record is 0001.
- Enter 0000 if there is no transaction code in the update records.

Columns 17-20
- Length of the transaction code in the update records.
- Maximum allowed is 0008.
- Enter 0000 if none.

Columns 22-25

- The start position within the update records where the data begins. The data within an update record must be in the same sequence as the 'processing map' AND each field (column) must be in DCLGEN format.
- Only those fields used on the screen should be in the update record, i.e., using DCLGEN output, remove any fields not used in the map and then rearrange in the sequence of the map.

Columns 27-30

- Length of the update data within the update record.
- This does not include transaction code if present. The maximum data length is 4000 bytes.

Columns 32-39

- The string of characters in the update record used to identify the record as an Insert transaction.
- If the update records have no transaction code and you want to allow inserts based on the 'Transaction Matrix' shown later, enter 'INSERT ' else leave blank.

Columns 41-48

- The string of characters in the update record used to identify the record as part of a Change transaction.
- There may be 1 or 2 records involved in a change transaction.
- If 2 records are involved, specify the order they appear in the update file in col 50-52.
- Both records must have the same transaction code string.
- If the update records have no transaction code and you want to allow changes based on the 'Transaction Matrix' shown later, enter 'CHANGE ' else leave blank.

Columns 50-52

- For Change records, containing two records per change, what is the sequence of the two rows for the change, i.e., old/new or new/old.
  'O N' = Old
followed by New
  'N O' = New
followed by Old.

  If specified the old record is used to search the record in the table to be changed and the new record is used to set the new values.

If only one record exists for a change, or changes are not in the update records, leave this blank.

Records to be changed are retrieved based on the following conditions:

a. If the table has a primary or unique key, the key columns only are used to retrieve the old record to be changed.

b. If the table has no primary or unique key, all of the columns given in the old input record are used to retrieve the record to be changed.

c. If there is only one record given for a change transaction, the table must contain a primary or unique key else the record will reject because it was not found or there will be no change made. This occurs based on the search logic specified above.

Columns 54-61

- The string of characters in the update record used to identify the record as a Delete transaction.
- If the update records have no transaction code and you want to allow deletes based on the 'Transaction Matrix' shown later, enter 'DELETE ' else leave blank.

Records to be deleted are retrieved based on the following conditions:

a. If the table has a primary or unique key, the key columns only are used to retrieve the old record to be deleted.

b. If the table has no primary or unique key, all of the columns given in the old input record are used to retrieve the record to be deleted.

The transaction input card is processed through the following decision rules to determine the exact update processing to use for a given input transaction.

Conditions	Rules	1	2	3	4	5	6	7	8	9	10	11	12
1	Txn. code in update record	N	N	N	N	N	N	N	N	Y	Y	Y	Y
2	Insert code (V=INSERT)	b	b	b	b	V	V	V	V	=Tx Cd.			N
3	Change code (V=CHANGE)	b	b	V	V	b	b	V	V		=Tx Cd.		N
4	Delete Code (V=DELETE)	b	V	b	V	b	V	b	V			=Tx Cd.	N

Actions		1	2	3	4	5	6	7	8	9	10	11	12
1	Apply Insert function	**X**				X				X			
2	Apply Delete function		X									X	
3	Apply Change function			X							X		
4	Error: cannot be both change & delete (need a txn. code)				X								
5	Apply Insert if not found, else apply Delete						X						
6	Apply Insert, if not found, else apply Change							X					
7	Error: update function is ambiguous								X				
8	Error: invalid transaction code												X

Legends: b = blank; Tx Cd. = transaction code in update record, V = INSERT, CHANGE or DELETE if no transaction code is in the update record.

OPTN - Control card - OPTN

This card is optional and it is used to specify certain processing options for the table to be updated.

Columns 01-04
- 'OPTN' Constant to identify this card.

Columns 10
- Table lock requirement
- ' ' - None
- 'E' - Exclusive lock
- 'S' - Shared lock

(The table lock is issued only for the starting table, i.e., all subsequent connected tables, if used in the screen, are not locked.)

Columns 12-16
- If commit is to be performed after processing some number of logical input records, enter the number desired.
- Enter zero or blank for commit at end of job.
- A logical record is an insert, a delete or a change whether the change transaction has 1 or 2 input records.

Columns 18-26
- If restart is to be performed, enter the number of input records to bypass.
- Blank or zero says restart is not required.
- Looking at the output of the original run, the correct value to enter here can be found in the last message generated regarding commits:
  - Commit performed at input record 'nnnnnn' or
  - All changes backed out after input record 'nnnnnn'.

Columns 28
- 'Y' if **TABLES/AS** control tables should be preloaded in main memory. (Do not use unless there is a large number of update transactions.)

Columns 30-34
- To terminate the job after some number of error conditions have been encountered, enter that Number.

```
//    JOB
//* ---------------------------------------------------------------- *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE
INTERNATIONAL
//* ---------------------------------------------------------------- *
//*
//* JCL: JCLUPD01
//*
//* DESCRIPTION: TABLES/AS DB2 TABLE BATCH
UPDATING USING SEQUENTIAL
//*            INPUT FILE.
//*
//* NOTES: REVIEW AND CHANGE EXTRACT FILE INPUT
DATASET. CHANGE
//*       TBLCARD CARD TO CORRECT TABLE NAME. THE
SYSTEM PARM
//*       ON THE DSN COMMAND MAY NEED TO CHANGE
ALSO.
//*
//* ---------------------------------------------------------------- *
//*     **********************************************************
//*     **   - CONTROL DD CARDS                        **
//*     **     CARD1 (REQUIRED)                        **
//*     **       COL 01-08 = CARD TYPE 'TABL'             **
//*     **       COL 10-17 = SCREEN NAME FOR EDITS
**
//*     **                 OR THE WORD '*DEFAULT' FOR     **
//*     **                 A DEFAULT SCREEN, CREATES A    **
//*     **                 NEW DEFAULT SCREEN EVERY TIME
**
//*     **       COL 19-26 = MAP NAME FOR THE TABLE
WITH    **
//*     **                 THE PROCESSING OPTIONS.  IF    **
//*     **                 NOT GIVEN, THE FIRST          **
//*     **                 PROCESSING RECORD             **
//*     **                 FOR THE GIVEN TABLE IS USED.  **
//*     **       COL 28-63 = TABLE NAME TO BE PROCESSED
**
//*     **     CARD2 (OPTIONAL)                        **
//*     **       COL 01-08 = CARD TYPE 'TRAN'             **
//*     **       COL 10    = DATA DELETE FLAG             **
//*     **             'Y'= DELETE ALL DATA FROM TABLE     **
//*     **                 BEFORE UPDATING               **
//*     **       COL 12-15 = UPDATE TRANSACTION CODE
**
//*     **                 START POSITION IN THE DATA     **
//*     **                 RECORD                    **
//*     **       COL 17-20 = UPDATE TRANSACTION CODE
**
//*     **                 LENGTH IN THE DATA RECORD      **
//*     **                 (THE LENGTH CANNOT BE GREATER
**
//*     **                  THAN 8 CHARACTERS)           **
//*     **       COL 22-25 = UPDATE DATA START
POSITION IN  **
//*     **                 THE DATA RECORD
**
//*     **       COL 27-30 = UPDATE DATA LENGTH
IN THE DATA **
//*     **                 RECORD                    **
//*     **       COL 32-39 = INSERT CODE, A
STRING OF ANY   **
//*     **                 CHARACTERS                  **
//*     **                 MUST BE 'INSERT' IF TXN.
CODE  **
//*     **                 IS NOT GIVEN                **
//*     **       COL 41-48 = CHANGE CODE, A
STRING OF ANY   **
//*     **                 CHARACTERS                  **
//*     **                 MUST BE 'CHANGE' IF TXN.
CODE  **
//*     **                 IS NOT GIVEN                **
//*     **       COL 50-52 = ORDER OF CHANGE
RECORDS, N & O **
//*     **                 OR O & N FOR OLD AND NEW
**
//*     **       COL 54-61 = DELETE CODE, A
STRING OF ANY   **
//*     **                 CHARACTERS                  **
//*     **                 MUST BE 'DELETE' IF TXN.
CODE  **
//*     **                 IS NOT GIVEN                **
//*     **     CARD3 (OPTIONAL)                          **
//*     **       COL 01-08 = CARD TYPE 'OPTN'
**
//*     **       COL 10    = IF TABLE IS TO BE
LOCKED     **
//*     **             'E'= EXCLUSIVE LOCK
**
//*     **             'S'= SHARED LOCK                **
//*     **       COL 12-16 = IF COMMIT IS TO BE
PERFORMED   **
//*     **                 AFTER UPDATING SO MANY
RECORDS **
//*     **                 ENTER COMMIT RECORD
COUNT     **
//*     **       COL 18-26 = IF RESTART IS TO BE
PERFORMED  **
//*     **                 ENTER RESTART RECORD
NO      **
//*     **       COL 28    = USE MAIN MEMORY TO
PRELOAD     **
//*     **                 TABLES/AS CONTROL
TABLES      **
//*     **             'Y'= USE MAIN MEMORY (DO
NOT USE    **
//*     **                    UNLESS YOU HAVE A LARGE NUMBER
**
//*     **                    OF INPUT TRANSACTIONS)        **
//*     **          COL 30-34 = IF JOB IS TO BE TERMINATED
**
//*     **                    AFTER A CERTAIN NUMBER OF      **
//*     **                    RECORDS IN ERROR, ENTER RECORD
**
//*     **                    COUNT                          **
//*     ************************************************************
//STA    EXEC TBLDB2DM,              ==> DB2 PROC
//         FUNC=BACHUPDT,            ==> FUNCTION CODE
//         TBLLIB='SSI.BATCH.LOADLIB' ==> TABLES/AS
BATCH LOADLIB
//ST1.STEPLIB DD
//           DD DSN=SSI.ONLINE.LOADLIB,DISP=SHR   ==>
ONLINE LOADLIB
//CONTROL  DD  *
TABL     *DEFAULT         SSI001.MAKE401
TRAN     Y 0000 0000 0001 0080 INSERT
OPTN     Y 00010 000000000 N
/*
//EDITS    DD  DSN=&EDITS,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(1,1)),
//         DCB=(LRECL=8000,BLKSIZE=8000,RECFM=FB)
//UPDATES  DD  DSN=USER.UPDATES,DISP=OLD,
//         DCB=(LRECL=4012,BLKSIZE=4016,RECFM=VB)
```

JCLUPD01 - Error Messages Generated

RETURN CODE ??

'-------- TAAS0501 RETURN CODE - XX , EXECUTION CANNOT CONTINUE ----'
There was an error encountered reading the SCREENDB control table. See TAAS0501 errors later in this section.

'-------- COMMIT PERFORMED AT INPUT RECORD= nnnnnn --------"
This message identifies that a commit occurred and up through which input record has been processed thru this commit. This is an informational message only.

RETURN CODE 8

'-------- ROLLBACK PERFORMED AT INPUT RECORD= nnnnnn --------'
'---ABENDING INFORMATION---- text,text,text,text,text'
'------ALL CHANGES BACKED OUT AFTER INPUT RECORD= nnnnn -----"
Some significant error occurred, probably exceeding maximum # of errors allowed, and a rollback has been performed. To restart the job from the last input record applied, use the value in message 'all changes backed out' as the value in the OPTN card columns 18-26.

'-------- UNABLE TO PERFORM ROLLBACK OPERATION, SQL CODE= -ccc----'
'--------- SQL ERROR MESSAGE= msg,msg,msg,msg,msg,msg ------'
See your DBA to resolve the problem. If needed, call SSI.

'-------- UNABLE TO PERFORM COMMIT OPERATION, SQL CODE= -ccc--------'
'--------- SQL ERROR MESSAGE= msg,msg,msg,msg,msg,msg ------'
See your DBA to resolve the problem. If needed, call SSI.

'------ INPUT FILE ENDS AT INPUT RECORD= nnnnnn RESTART 'CANNOT BE PERFORMED ------'
The input file ran out of records before reaching the restart record number specified in the OPTN card, cols 18-26. Check to insure you entered the correct value in the OPTN card and that you are using the correct input file

'----CHANGE ROWS MUST BE INDICATED BY O,N(OLD,NEW) OR N,O(NEW, OLD) OR ABSENT ------'
In TRAN card, if col 50= N or O, col 52 must be the opposite. Else, both columns must be blank, i.e., there is only 1 record per change transaction in the update file.

'----INSERT CODE MUST BE - INSERT --------'
If transaction code start position in the TRAN card cols 12-15 is zero or spaces, if there is a value in the insert code field (col 32-39) it must be 'INSERT'.

'----CHANGE CODE MUST BE - CHANGE --------'
If transaction code start position in the TRAN card cols 12-15 is zero or spaces, if there is a value in the insert code field (col 41-48) it must be 'CHANGE'.

'----DELETE CODE MUST BE - DELETE --------'
If transaction code start position in the TRAN card cols 12-15 is zero or spaces, if there is a value

in the insert code field (col 54-61) it must be 'DELETE'.

'----PARM. VALUE FOR THE COMMIT COUNT IS INVALID ------'
Cols 18-26 in the OPTN card are not spaces, zeros or numeric.

'----PARM. VALUE FOR THE RESTART RECORD NO IS INVALID ------'
Cols 18-26 in the OPTN card are not spaces, zeros or numeric.

'----PARM. VALUE FOR THE ERROR COUNT IS INVALID ------'
Cols 18-26 in the OPTN card are not spaces, zeros or numeric.

'----INPUT FILE ENDS AT INPUT RECORD= nnnnnn INVALID ORDER TO CHANGE RECORDS'
Before detecting the 2nd change record, end of file was reached, i.e., the old or new record was missing field (col 54-61) it must be 'DELETE'

'----ERROR, THE UPDATE FUNCTION IS AMBIGUOUS, CANNOT BE INSERT, CHANGE AND DELETE, INPUT RECORD BEING PROCESSED WAS= nnnnnn --'
Transaction start position is spaces or zero and there are values in Insert, Change and Delete code strings in TRAN card.

'----ERROR, THERE IS NO SPECIFICATION OF WHAT UPDATE FUNCTION TO USE, INPUT RECORD BEING PROCESSED WAS= nnnnnn'
Transaction start position is spaces or zero and there are NO values in Insert, Change and Delete code strings in TRAN card.

'----ERROR, THE UPDATE FUNCTION CANNOT BE BOTH THE CHANGE AND DELETE, INPUT RECORD BEING PROCESSED WAS = nnnnnn --'
Transaction start position is spaces or zero and there are values in the Change and Delete code strings in TRAN card. Check the matrix to determine the valid entries.

'--------TRANSACTION CODE START & LENGTH ARE INVALID ------'
Both fields must be spaces or zeros as input or both fields must contain a number greater than zero.

'--------DATA START & LENGTH ARE INVALID-------'
Both fields must be spaces or zeros as input or both fields must contain a number greater than zero.

'----TRANSACTION START & DATA START BOTH MUST BE ZERO OR > ZERO'
Both fields must be spaces or zeros as input or both fields must contain a number greater than zero.

'----UNABLE TO DELETE RECORDS FROM TABLE , tblname SQL CODE- -nnn'
'--------- SQL ERROR MESSAGE= msg,msg,msg,msg,msg,msg ------'
A DB2 error occurred trying to delete the table rows before starting the update process. Check with your DBA to resolve. If needed, contact SSI.

'----SCREEN MAP RECORD IS REQUIRED ------'
Make certain the screen name is valid in the TABL card in col 10-17. Use **TABLES/AS** list screen function to insure screen exists. If

everything is correct, contact SSI. The extract of TABLES/AS data from he Screendb is incomplete

'----SCREEN PROCESSING RECORD IS REQUIRED --------'
Specify the map name that should be used in the TABL card cols 19-26. The screen being used has more than 1 map for the table. Make certain the map name corresponds to a processing map. If this error recurs, contact SSI. The extract of **TABLES/AS** data from the Screendb is incomplete or SCRRCD record extracted is not a processing map for the table.

'----SCREEN EDIT RECORD IS REQUIRED --------'
If this occurs, go to **TABLES/AS** and Prepare the Screen using Option 5. Then re-run the job. If this error recurs, contact SSI. The extract of TABLES/AS data from the Screendb is incomplete.

'----SCREEN MAPPING RECORDS ARE REQUIRED --------'
Go to **TABLES/AS** Option 3 Record Mapping and insure that there is a MAP in the correct table name and map name. If not, either create one or re-run the job with the correct table and map names. Prepare the screen if you made any changes. If this error recurs, contact SSI. The extract of **TABLES/AS** data from the Screendb is incomplete.

TAAS0501 Errors

'----TABLCARD MUST CONTAIN A TABLE NAME------'
Column 28-63 must not be blank in the TABL card

'----TABLCARD MUST CONTAIN A SCREEN NAME OR THE WORD DEFAULT----'
Column 10-17 must contain valid screen name or the keyword '*DEFAULT'.

'----A DEFAULT SCREEN COULD NOT BE CREATED----'
- Too many fields in the table. Current limit = 120
- Too many characters to fit on a screen.
- Use **TABLES/AS** and build a screen with a subset of the data fields using the SEL option. Reformat your input Tx's if needed and then retry Note: Use the name of the screen you created not '*DEFAULT'.

'----SCREEN= "screenname" DOES NOT CONTAIN TABLE MAP-----'
No map has been defined for the screen being used. Go into **TABLES/AS** and use Options 3, 4 & 5 to define a map, define one as a processing map and prepare the screen. Then retry.

'----SCREEN="screenname" DOES NOT CONTAIN TABLE EDITS----'
No mapping was defined for the screen & table or all edits have been deleted in **TABLES/AS**. Go into **TABLES/AS** and use Options 3, 2, 4 & 5 to correct the problem. Then retry.

'----SCREEN="screenname" DOES NOT CONTAIN TABLE PROCESSING----'

No map has been defined as a processing map. Go into **TABLES/AS** and use Options 4 & 5 to correct the problem. Then retry.

'----SCREEN="screenname" DOES NOT CONTAIN SCREEN FORMAT RECORD----'
No screen format exists in the SCREENDB for the screen name specified. Insure that the screen name in the TABL card is correct. Insure that the plan is pointing to the correct set of SSI control tables (AUTHID). If all else fails, contact SSI.

'----SCREEN="screenname" DOES NOT CONTAIN SCREEN FIELD NAME RECORD--'
This should never occur. Contact SSI.

TAAS0501 Errors (Continued)

'----SCREEN="screenname" MAP NAME="xxxxxxxx" DOES NOT CONTAIN MAPPING RECORDS----'
The map name specified in the TABL card is not valid for this screen. There are no records in the SCRRCD control table. Check **TABLES/AS** Options 3, 4 & 5 to correct. Then rerun.

'----SCREEN="screenname" FIELD NAME = "xxxxxxxx" IS NOT FOUND IN THE NAME RECORD----'
Column names in the DB2 may have changed since this screen was created to process the table. Check the current DB2 field names against those in **TABLES/AS** Option 3. If different, delete the map with Option 3 and then perform Options 3, 4, 2 & 5 to correct. Then retry. If still incorrect, contact SSI.

'----TABLE="tablename" IS NOT FOUND IN DB2----'
- The table name in the TABL card is not valid.
- The table in a previously defined map may have been deleted.
- Perhaps the wrong DB2 subsystem is being looked at.
- Perhaps an incorrect set of SSI Control tables are being pointed to by the DB2 plan.

'----TABLE="tablename" CONTAINS ONE OR MORE UNSUPPORTED COLUMN TYPE IN DB2---'
- Create a **TABLES/AS** screen using the SEL function to exclude the unsupported column types, revise your TX file to correspond to the resultant map and retry.
- Unsupported column types are: FLOAT and ALL GRAPH types.

'----TABLE="tablename" CONTAINS MORE THAN 120 COLUMNS, PROCESSING IS TERMINATED----'
If you can create a DB2 view with less than 120 columns, you could use the view.

'----TABLE="tablename" DB2 CATALOGUE READ ERROR SQL CODE=xxxx--'
Determine the cause with your DBA. If still a problem, contact SSI.

JCLMASSC - Mass Change Batch Update

This utility uses DB2 plan TBLASBCH.

Mass Change Batch Update creates a batch file of mass change activity via step A which is input into step B, the generalized batch update function.

The TABLES/AS screen to be used in this job must already exist in TABLES/AS prior to starting this job. That screen name is entered into the CONTROL DD statement in STA and STB.

STA extracts the necessary information from the TABLES/AS stored data for the named screen. This data specifies the columns of the table that are used on the screen. It then reads a DB2 table creating change transactions, old followed by new, for each table row containing the specified values in the column name used in the control card. You may process only one column with one set of field values per run. DB2 rows to change are selected based on 'OLDValue'. The specified 'NEWValue' will be used to replace the old value in the DB2 row.

STB actually performs the edit verification and batch update. See the prior write-up (JCLUPD01) for an explanation of this process.

> STB Note:
> Normally, on the TABL control card needs to be changed. Enter the name of the screen and the name of the table. All other control card information is valid. If you are doing a re-run, see write-up for JCLUPD01 for control card entries to restart.

In each step, the table name and screen name specified must be consistent for a given run.

```
//       JOB
//SSIPROC JCLLIB ORDER=SSI.ONLINE.JCL   ==> TBLDB2DM
PROCEDURE LIBRARY
//* --------------------------------------------------------------- *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE
INTERNATIONAL
//* --------------------------------------------------------------- *
//*
//* JCL: JCLMASSC
//*
//* DESCRIPTION: TABLES/AS DB2 TABLE MAINTENANCE FOR
MASS CHANGE.
//*      NOTE:  AN EXTRACT FILE IS CREATED FOR PROCESSING
THROUGH
//*            TABLES/AS BATCH UPDATE USING 'CHANGE'
FUNCTION.
//*            THE ORDER OF INPUT RECORDS IN THE EXTRACT
FILE IS
//*            OLD AND NEW (O&N). MAKE SURE YOU ARE POINTING
//*            TO THE LIBRARY THAT CONTAINS THE 'TBLDB2DM'
PROC.
//*
//* NOTES: IN EACH STEP,
//*      CHANGE CONTROL CARDS TO CORRECT TABLE NAME &
SCREEN NAME.
//*      THE SYSTEM PARM IN THE DSN COMMAND STATEMENTS
MAY NEED
//*      TO CHANGE ALSO. ENTER THE TBLLIB PARAMETER
POINTING TO THE
//*      BATCH LOADLIB. ENTER THE NAME OF THE TABLES/AS
LIBRARY
//*      CONTAINING THE ON-LINE MODULES.
//*
//*
//* --------------------------------------------------------------- *
//*
//*     *********************************************************
//*     **                                        **
//*     **   - UPDATES  DD IS THE OUTPUT AND IT CONTAINS THE
**
//*     **       CHANGE RECORDS IN THE OLD&NEW
SEQUENCE(O&N)**
//*     **                                        **
//*     **   - CONTROL DD CARDS                      **
//*     **     CARD1 (REQUIRED)                      **
//*     **       COL 01-08 = CARD TYPE 'SCRE'             **
//*     **       COL 10    = SCREEN NAME FOR EDIT         **
//*     **     CARD2 (REQUIRED)                      **
//*     **       COL 01-08 = CARD TYPE 'TABL'             **
//*     **       COL 10    = TABLE NAME TO UPDATE         **
//*     **     CARD3 (REQUIRED)                      **
//*     **       COL 01-08 = CARD TYPE 'COLU'             **
//*     **       COL 10    = COLUMN NAME TO CHANGE        **
```

```
//*     **     CARD4 (REQUIRED)                      **
//*     **       COL 01-08 = CARD TYPE 'OLDV'
//*     **       COL 10    = OLD VALUE TO CHANGE F
**
//*     **     CARD5 (REQUIRED)                      **
//*     **       COL 01-08 = CARD TYPE 'NEWV'
//*     **       COL 10    = NEW VALUE TO CHANGE T
**
//*     **                                        **
//*     **   - SYSTSIN DD CARD                       **
//*     **       CHECK SYSTEM PARM AND PLAN NA
**
//*     *********************************************************
//*
//* ---------------------------------------------------------------
//STA    EXEC TBLDB2DM,
//       TBLLIB='SSI.BATCH.LOADLIB',   ==> TABLE
BATCH LOADLIB
//       FUNC=MASSCHNG               ==> CREATE M
CHANGE FILE
//ST1.STEPLIB  DD
//           DD  DSN=SSI.ONLINE.LOADLIB,DISP=SHI
//UPDATES  DD  DSN=&&MASSC,DISP=(NEW,PASS
//       DCB=(LRECL=4012,BLKSIZE=4016,RECFM=
//       UNIT=SYSDA,SPACE=(TRK,(2,1))
//EDITS    DD  DSN=&&TABLEDEF,DISP=(NEW,PAS
//       UNIT=SYSDA,SPACE=(CYL,(1,1)),
//       DCB=(LRECL=8000,BLKSIZE=8000,RECFM=
//CONTROL  DD  *
SCREEN   SCRNAME
TABLE    SSI006.CARCL41
COLUMN   MILEAGE_LIMIT
OLDVALUE 70
NEWVALUE 71
/*
//* ---------------------------------------------------------------
//*     *********************************************************
//*     **   - CONTROL DD CARDS
//*     **     CARD1 (REQUIRED)                      **
//*     **       COL 01-08 = CARD TYPE 'TABL'
//*     **       COL 10-17 = SCREEN NAME FOR EDIT
**
//*     **       COL 19-26 = MAP NAME FOR THE TAE
WITH    **
//*     **               THE PROCESSING OPTIONS.  I
//*     **               NOT GIVEN, THE FIRST         **
//*     **               PROCESSING RECORD
//*     **               FOR THE GIVEN TABLE IS USE
//*     **       COL 28-63 = TABLE NAME TO BE
PROCESSED    **
//*     **     CARD2                                 **
//*     **       COL 01-08 = CARD TYPE 'TRAN'
//*     **       COL 10    = DATA DELETE FLAG
//*    **          'N'= DELETE ALL DATA FROM TABLE    **
//*    **               BEFORE UPDATING               **
//*    **       COL 12-15 = UPDATE TRANSACTION CODE       **
//*    **               START POSITION IN THE DATA    **
//*    **               RECORD                        **
//*    **       COL 17-20 = UPDATE TRANSACTION CODE       **
//*    **               LENGTH IN THE DATA RECORD     **
//*    **               (THE LENGTH CANNOT BE GREATER  **
//*    **                THAN 8 CHARACTERS)           **
//*    **       COL 22-25 = UPDATE DATA START POSITION IN  **
//*    **               THE DATA RECORD               **
//*    **       COL 27-30 = UPDATE DATA LENGTH IN THE DATA **
//*    **               RECORD                        **
//*    **       COL 41-48 = CHANGE CODE, A STRING OF ANY   **
//*    **               CHARACTERS                    **
//*    **               MUST BE 'CHANGE' IF TXN. CODE  **
//*    **               IS NOT GIVEN                  **
//*    **       COL 50-52 = ORDER OF CHANGE RECORDS, N & O **
//*    **               OR O & N FOR OLD AND NEW      **
//*    **     CARD3 (OPTIONAL)                        **
//*    **       COL 01-08 = CARD TYPE 'OPTN'              **
//*    **       COL 10    = IF TABLE IS TO BE LOCKED       **
//*    **          'E'= EXCLUSIVE LOCK                **
//*    **          'S'= SHARED LOCK                   **
//*    **       COL 12-16 = IF COMMIT IS TO BE PERFORMED   **
//*    **               AFTER UPDATING SO MANY RECORDS **
//*    **               ENTER COMMIT RECORD COUNT      **
//*    **       COL 18-26 = IF RESTART IS TO BE PERFORMED  **
//*    **               ENTER RESTART RECORD NO        **
//*    **       COL 28    = USE MAIN MEMORY TO PRELOAD     **
//*    **               TABLES/AS CONTROL TABLES       **
//*    **          'Y'= USE MAIN MEMORY (DO NOT USE    **
//*    **               UNLESS YOU HAVE A LARGE NUMBER **
//*    **               OF INPUT TRANSACTIONS)         **
//*    **       COL 30-34 = IF JOB IS TO BE TERMINATED     **
//*    **               AFTER A CERTAIN NUMBER OF      **
//*    **               RECORDS IN ERROR, ENTER RECORD **
//*    **               COUNT                         **
//*    ***********************************************************
//*
//* ------------------------------------------------------------------ *
//STB    EXEC TBLDB2DM,              ==> DB2 PROC
//        TBLLIB='SSI.BATCH.LOADLIB',   ==> TABLES BATCH
LOADLIB
//        FUNC=BACHUPDT              ==> DUMMY FUNCTION CODE
//ST1.STEPLIB  DD
//           DD  DSN=SSI.ONLINE.LOADLIB,DISP=SHR
//CONTROL  DD  *
TABL     PA             SSI006.CARCL41
TRAN     N 0000 0000 0000 0000         CHANGE   O&N
OPTN       00000 000000000 N
/*
//EDITS    DD  DSN=&EDITS,DISP=(NEW,PASS),
//        UNIT=SYSDA,SPACE=(CYL,(1,1)),
//        DCB=(LRECL=8000,BLKSIZE=8000,RECFM=
//UPDATES  DD  DSN=&&MASSC,DISP=(OLD,PASS
//        DCB=(LRECL=4012,BLKSIZE=4016,RECFM=
//SYSABOUT DD  SYSOUT=*
//SYSABEND DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
```

JCLMASSD - Mass Delete Batch Update

This utility uses DB2 plan TBLASBCH.

Mass Delete Batch Update creates a batch field of mass delete activity via step A which are input into step B, the generalized batch update function.

The TABLES/AS screen to be used in this job must already exist in TABLES/AS prior to starting this job. That screen name is entered into the CONTROL DD statement in STA and STB.

STA extracts the necessary information from the TABLES/AS stored data for the named screen. This data specifies the columns of the table that are used on the screen. It then reads a DB2 table creating delete transactions for each table row containing the specified value in the column name used in the control card. You may process only one column with one field value per run.

STB actually performs the edit verification and batch update. See the prior write-up (JCLUPD01) for an explanation of this process.

STB Note:

Normally, only the TABL control card needs to be changed. Enter the name of the screen and the name of the table. All other control card information is valid. If you are doing a re-run, see write-up for JCLUPD01 for control card entries to restart.

In each step, the table name and screen name specified must be consistent for a given run.

```
//        JOB
//SSIPROC JCLLIB ORDER=SSI.ONLINE.JCL    ==> TBLDB2DM
PROCEDURE LIBRARY
//* --------------------------------------------------------------- *
//*      TABLES SYSTEM - SPECIALIZED SOFTWARE
INTERNATIONAL
//* --------------------------------------------------------------- *
//*
//* JCL: JCLMASSD
//*
//* DESCRIPTION: TABLES/AS DB2 TABLE MAINTENANCE FOR
MASS DELETE.
//*      NOTE:  AN EXTRACT FILE IS CREATED FOR PROCESSING
THROUGH
//*            TABLES/AS BATCH UPDATE USING 'DELETE'
FUNCTION.
//*            MAKE SURE YOU ARE POINTING TO THE LIBRARY
THAT
//*            CONTAINS THE 'TBLDB2DM' PROC.
//*
//* NOTES: IN EACH STEP,
//*       CHANGE CONTROL CARD TO CORRECT TABLE NAME &
SCREEN NAME.
//*       THE SYSTEM PARM IN THE DSN COMMAND STATEMENTS
MAY NEED
//*       TO CHANGE ALSO. TBLLIB PARAMETER SHOULD POINT
TO YOUR
//*       LOADLIB CONTAINING THE TABLES/AS BATCH MODULES.
ENTER THE
//*       LOADLIB CONTAINING THE TABLES/AS ON-LINE
MODULES.
//*
//*
//* --------------------------------------------------------------- *
//*
//*     ********************************************************
//*     **                                        **
//*     **   - UPDATES  DD IS THE OUTPUT AND IT CONTAINS THE
**
//*     **        DELETE RECORDS                  **
//*     **                                        **
//*     **   - CONTROL DD CARDS                   **
//*     **     CARD1 (REQUIRED)                   **
//*     **        COL 01-08 = CARD TYPE 'SCRE'           **
//*     **        COL 10    = SCREEN NAME TO EDIT        **
//*     **     CARD2 (REQUIRED)                   **
//*     **        COL 01-08 = CARD TYPE 'TABL'           **
//*     **        COL 10    = TABLE NAME TO UPDATE       **
//*     **     CARD3 (REQUIRED)                   **
//*     **        COL 01-08 = CARD TYPE 'COLU'           **
//*     **        COL 10    = COLUMN NAME TO DELETE FOR     **
//*     **     CARD4 (REQUIRED)                   **
//*     **        COL 01-08 = CARD TYPE 'OLDV'           **
```

```
//*     **        COL 10    = OLD VALUE TO DELETE B
**
//*     **                                        **
//*     **   - SYSTSIN DD CARD                    **
//*     **        CHECK SYSTEM PARM AND PLAN NA
**
//*     ********************************************************
//*
//* ---------------------------------------------------------------
//STA    EXEC TBLDB2DM,
//       TBLLIB='SSI.BATCH.LOADLIB',   ==> TABLE
BATCH LOADLIB
//       FUNC=MASSDLET              ==> CREATE M
DELETE FILE
//ST1.STEPLIB  DD
//          DD  DSN=SSI.ONLINE.LOADLIB,DISP=SH
//UPDATES  DD  DSN=&&MASSD,DISP=(NEW,PASS
//       DCB=(LRECL=4012,BLKSIZE=4016,RECFM=
//       UNIT=SYSDA,SPACE=(TRK,(2,1))
//EDITS    DD  DSN=&&TABLEDEF,DISP=(NEW,PAS
//       UNIT=SYSDA,SPACE=(CYL,(1,1)),
//       DCB=(LRECL=8000,BLKSIZE=8000,RECFM=
//CONTROL  DD  *
SCREEN   SCRNAME
TABLE    SSI006.CARCL41
COLUMN   MILEAGE_LIMIT
OLDVALUE 71
/*
//*
//* ---------------------------------------------------------------
//*     ********************************************************
//*     **   - CONTROL DD CARDS
//*     **     CARD1 (REQUIRED)                   **
//*     **        COL 01-08 = CARD TYPE 'TABL'
//*     **        COL 10-17 = SCREEN NAME FOR EDIT
**
//*     **                OR THE WORD '*DEFAULT' FO
//*     **                A DEFAULT SCREEN, CREATE
**
//*     **                NEW DEFAULT SCREEN EVER
**
//*     **        COL 19-26 = MAP NAME FOR THE TA
WITH    **
//*     **                THE PROCESSING OPTIONS.  I
//*     **                NOT GIVEN, THE FIRST         *
//*     **                PROCESSING RECORD
//*     **                FOR THE GIVEN TABLE IS USE
//*     **        COL 28-63 = TABLE NAME TO BE
PROCESSED    **
//*     **     CARD2 (OPTIONAL)                   **
//*     **        COL 01-08 = CARD TYPE 'TRAN'
//*     **        COL 10    = DATA DELETE FLAG
//*    **         'N'= DELETE ALL DATA FROM TABLE    **
//*    **              BEFORE UPDATING          **
//*    **      COL 12-15 = UPDATE TRANSACTION CODE      **
//*    **              START POSITION IN THE DATA    **
//*    **              RECORD                **
//*    **      COL 17-20 = UPDATE TRANSACTION CODE      **
//*    **              LENGTH IN THE DATA RECORD     **
//*    **              (THE LENGTH CANNOT BE GREATER  **
//*    **               THAN 8 CHARACTERS)        **
//*    **      COL 22-25 = UPDATE DATA START POSITION IN  **
//*    **              THE DATA RECORD          **
//*    **      COL 27-30 = UPDATE DATA LENGTH IN THE DATA **
//*    **              RECORD                **
//*    **      COL 54-61 = DELETE CODE, A STRING OF ANY   **
//*    **              CHARACTERS             **
//*    **              MUST BE 'DELETE' IF TXN. CODE  **
//*    **              IS NOT GIVEN            **
//*    **    CARD3 (OPTIONAL)                  **
//*    **      COL 01-08 = CARD TYPE 'OPTN'         **
//*    **      COL 10    = IF TABLE IS TO BE LOCKED      **
//*    **           'E'= EXCLUSIVE LOCK          **
//*    **           'S'= SHARED LOCK            **
//*    **      COL 12-16 = IF COMMIT IS TO BE PERFORMED   **
//*    **              AFTER UPDATING SO MANY RECORDS **
//*    **              ENTER COMMIT RECORD COUNT     **
//*    **      COL 18-26 = IF RESTART IS TO BE PERFORMED  **
//*    **              ENTER RESTART RECORD NO      **
//*    **      COL 28    = USE MAIN MEMORY TO PRELOAD     **
//*    **              TABLES/AS CONTROL TABLES      **
//*    **           'Y'= USE MAIN MEMORY (DO NOT USE    **
//*    **              UNLESS YOU HAVE A LARGE NUMBER **
//*    **              OF INPUT TRANSACTIONS)       **
//*    **      COL 30-34 = IF JOB IS TO BE TERMINATED     **
//*    **              AFTER A CERTAIN NUMBER OF     **
//*    **              RECORDS IN ERROR, ENTER RECORD **
//*    **              COUNT                 **
//*    **********************************************************
//*
//* ----------------------------------------------------------------- *
//STB    EXEC TBLDB2DM,              ==> DB2 PROC
//       TBLLIB='SSI.BATCH.LOADLIB',   ==> TABLES/AS BATCH
LOADLIB
//       FUNC=BACHUPDT               ==> DUMMY FUNCTION CODE
//ST1.STEPLIB  DD
//          DD  DSN=SSI.ONLINE.LOADLIB,DISP=SHR
//CONTROL  DD  *
TABL     PA              SSI006.CARCL41
TRAN     N 0000 0000 0000 0000                  DELETE
OPTN       00000 000000000 N
/*
//EDITS    DD  DSN=&EDITS,DISP=(NEW,PASS),
//       UNIT=SYSDA,SPACE=(CYL,(1,1)),
//       DCB=(LRECL=8000,BLKSIZE=8000,RECFM=FB)
```

```
//UPDATES  DD  DSN=&&MASSD,DISP=(OLD,PASS
//       DCB=(LRECL=4012,BLKSIZE=4016,RECFM=
//SYSABOUT DD  SYSOUT=*
//SYSABEND DD  SYSOUT=*
```