INTERCOMM

TABLE FACILITY

ISOGON CORPORATION
330 Seventh Avenue · New York, New York 10001

LICENSE: INTERCOMM TELEPROCESSING MONITOR

Table Facility

Publishing History

Publication	Date	Remarks
First Edition	August 1993	Documenting the new Table Facility and its usage. (For Release 10 only.)

PREFACE

Intercomm is a state-of-the-art teleprocessing monitor system executing on the IBM System/370 and System/390 family of computers and operating under the control of IBM Operating Systems (XA and ESA). Intercomm monitors the transmission of messages to and from terminals, concurrent message processing, centralized access to I/O files, and the routine utility operations of editing input messages and formatting output messages, as required.

This document describes the Table Facility and its usage in online and in batch mode. The Table Facility may be used to create user tables in core storage above the 16M line (31-Amode) to use as a scratch pad area or to hold data records. Tables may be kept and modified or reused during the life of one Intercomm or batch mode job execution. They may not be shared across address spaces (regions).

The Table Facility may also be used by Intercomm system programs, therefore all table entry processing is under control of the Table Facility to prevent storage destruction.

In conjunction with the use of this document, the reader should consult the following Intercomm publications:

- Operating Reference Manual
- Basic Systems Macros
- Programmers Guides (COBOL, BAL, PL/1)

INTERCOMM PUBLICATIONS

GENERAL INFORMATION MANUALS

Concepts and Facilities

Planning Guide

APPLICATION PROGRAMMERS MANUALS

Assembler Language Programmers Guide

COBOL Programmers Guide

PL/1 Programmers Guide

SYSTEM PROGRAMMERS MANUALS

Basic System Macros

BTAM Terminal Support Guide

Installation Guide

Messages and Codes

Operating Reference Manual

System Control Commands

CUSTOMER INFORMATION MANUALS

Customer Education Course Catalog

Technical Information Bulletins

User Contributed Program Description

FEATURE IMPLEMENTATION MANUALS

Autogen Facility

ASMF Users Guide

DBMS Users Guide

Data Entry Installation Guide

Data Entry Terminal Operators Guide

Dynamic Data Queuing Facility

Dynamic File Allocation

Extended Security System

File Recovery Users Guide

Generalized Front End Facility

Message Mapping Utilities

Multiregion Support Facility

Page Facility

Store/Fetch Facility

SNA Terminal Support Guide

Table Facility

TCAM Support Users Guide

Utilities Users Guide

EXTERNAL FEATURES MANUALS

SNA LU6.2 Support Guide

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Table Facility is a table build, management and processing facility that can be used by application programs running under control of Intercomm, or in batch mode. Implementation of the Table Facility is designed to relieve application programs of table handling code. The user will only need to supply the data entries to be put in a table along with the unique name to be assigned to the table and the fixed length of the entries in a specific table. The Table Facility management and processing software executes in the Intercomm region below the 16M line, however, all table control blocks and table data areas will be in storage acquired above the 16M line to relieve program storage requirements. No data sets or file I/O are required by the Table Facility.

1.2 EXTERNAL DESIGN OVERVIEW

The Table Facility provides an application program the software (via callable interfaces) to manage temporary (for the life of one Intercomm system execution) table creation (building) and access. Each table will have a unique (program-specified) name and there may be as many tables as storage constraints above the 16M line under IBM's MVS/XA or MVS/ESA allow. The maximum size of one table is approximately 16M. The entry size (up to 32767 bytes) for a specific table is specified when the table is built, as also (optionally) are the entry's key offset and size within an entry. If table entries contain keys, the table may optionally be sorted at program request before subsequent access. After a table is built, a specific entry in the table may be randomly accessed via key (if any) or relative entry number, or entries may be retrieved in sequential or reverse order. Existing entries may be updated or deleted at program request after the table is built. Deletion of tables when no longer needed is a program responsibility, except if an on-line program should program check or time out while building a table. In the latter cases, Intercomm application program purge processing will delete a table being built, or close a table being accessed for update, as needed. However, if the address space abends or is closed down, MVS will free all table storage.

1.3 INTERCOMM SYSTEM INTERFACE

The system Table Facility interface consists of one new program called INTTABLE which contains the entry points for the table management interface routines to be called by user application programs. The INTTABLE module executes below the 16M line and handles all table management and control block processing, including mode-switching for access to table control blocks (TFCBs) and to build and process table data areas acquired in storage above the 16M line. The INTTABLE module also handles Table Facility processing statistics gathering (see Chapter 5) and optional snaps of table entry activity (see Chapter 4).

Parameters are provided on the system SPALIST macro for the user to specify (at Table Facility installation) the initial and increment sizes of the TFCB (Table Facility Control Block) area and of each table entries area (the default is 4K (4096 bytes - the IBM MVS page size)), and the system-wide number of the MVS subpool (from 1 to 127) to use for the table areas (to group them together) if the default of subpool 0 (zero) is not desired. See Chapter 4.

System statistics processing modules have been updated to print 31-Amode storage request and usage totals, and to print Table Facility statistics for tuning the SPALIST parameters. See Chapter 5 for details.

Relevant system command processing modules have been modified for Table Facility information display. See Chapter 5 for details.

1.4 USER PROGRAM INTERFACE

A user application program accesses the Table Facility via a CALL to the desired INTTABLE interface entry point. The entry point names and usage are given in Chapter 2, along with suggested Table Facility access scenarios. Detailed descriptions of the called routines, the parameters to pass, and the return codes, are given in Chapter 3.

One on-line program thread can concurrently access up to 255 tables. There is no restriction for a batch mode program. See Chapter 6 for batch mode processing and linkedit.

1.5 DEBUGGING AND ERROR INFORMATION

Where possible, an error return code after a CALL to the Table Facility is made to indicate programming errors and to prevent program loops. Should a program check or timeout occur in a program accessing the Table Facility, Appendix A gives debugging information and pertinant control block data. Should the Table Facility encounter an unrecoverable error (usually due to storage overlay), it will force a program check: error codes issued by the Table Facility and their causes are given in Appendix B.

CHAPTER 2

TABLE USAGE

2.1 GENERAL USAGE

Tables can be used by system or user application programs for temporary scratch pad areas, for user statistics gathering, for record gathering for subsequent analysis and/or print, or for user tables and data areas (instead of the user SPA, for example), or to save terminal work areas. Unique table names (up to 16 bytes in length) are ensured.

Data entries in a table are fixed length. The entry length for a table is specified when a table is built, and can be from 1 to 32767 bytes. If variable length entries are desired, the program must process such entries by specifying the actual entry length within the entry (or in the first halfword of the entry) and padding the unused remainder of the program entry area with binary zeroes (low-values) or blanks, as appropriate. The maximum entry size is specified when the table is built and that maximum size is used by INTTABLE for moving the entry between the table and the caller's entry data area. The program retrieving the entry must be aware of a variable length entry and process accordingly.

Entries may have keys from 1 to 256 bytes in length (must be the same length and in the same place in each entry). The key must be entirely located within the first 256 bytes of each entry. The key size and offset is also defined when the table is built. Keyed entries may be sorted (at program request) for subsequent access by key. INTTABLE uses the system sort and binary search routines (INTSORT and BINSRCH) for sorting keyed tables and for locating an entry by key in a pre-sorted table. Keyed and non-keyed entries within a table may not be intermixed if the table is to be sorted.

For both keyed and non-keyed tables, entries may be created when the table is originally built, or may be added (appended) to an existing table. New entries may not be inserted in a table. Existing entries may be updated or deleted. For sorted keyed tables, if an entry is deleted in the middle of the table, or if an update changes the key area, or if new entries are added, the table is flagged to be resorted when access ends (table is closed), and further access by key is prevented until the table is resorted. A user program can, however, request the resort if further access by key is desired.

Simultaneous access for update (or add or delete) to the same table from multiple programs (program threads) is not allowed. Shared program access, for retrieval of entries only, is permitted. The type of access to an existing table is program-specified when the table is opened for access. Access for retrieval only may not be changed to access for update (or add or delete) without first closing the table (ending build or retrieval access) and then reopening it for update. Thus, entry integrity is assured.

To update or delete an existing entry, that entry must first be retrieved for update. Deleted entries in a keyed table are discarded when the table is resorted. Deleted entries are initialized to X'FF' (high-values) and are not accessible except by relative entry number (automatically skipped during sequential retrieval). If the first entry in a sorted table is deleted, the subsequent entries are shifted over it, the last entry is then deleted and resorting is not needed. If the last entry in a table is deleted, it is no longer accessible (even by entry number) and resorting is not needed.

2.2 INTERNAL PROCESSING

Because tables are in storage above the 16M line, they are infinitely expandable (up to 16M). When a table is built (first created), an initial table area is acquired. The size of the initial area is a multiple of 1K as specified via a SPALIST parameter when the Table Facility is installed (default is 4K or 4096 bytes). The increment for expanding a table (multiple of 1K) is also specified on the SPALIST macro. The maximum size of one table, therefore, is actually 16M less the increment value. The size of the initial and increment values must be at least as large as the maximum expected entry length. When a table entry area is expanded, a new area of existing size plus the increment size is acquired, the existing entries are copied to the new area, the old area is freed, then the new entry is added (in the new area). If entries at the end of a table are deleted (also if deleted entries exist after a sort), then any increment(s) is freed if possible. Thus, initial and increment default sizes are set at 4K to use MVS page sizes. Note that if all entries in a table are deleted, the entire table area is freed. Statistics are kept on table storage, table expansions, the largest table, the maximum number of entries created in a table, and the average table size, for tuning the SPALIST parameters.

On the first call to build a table (after system startup) an area of storage above the 16M line will be acquired to hold the Table Facility Control Blocks (TFCB's). Each control block is 64-bytes in length and contains information about each table being built or accessed, such as table name, entry length, key length and offset (if any), number of entries (incremented as table entries added), address of the table entry area, size of the table entry area, a pointer to the last added/valid entry in the table area, internal key of the program creating or updating the table (for program purge processing - if needed), and other control flags and fields. The TFCB area is in system-controlled 31-Amode storage and will be of an initial (user-specified in the SPA) size which will be automatically expanded (relocated) as needed. The address of this area is placed in the System Parameter Area (SPA).

When a table is closed with delete, the table entry area is freed, and the TFCB in the system TFCB area is cleared and put on a chain of free TFCB's in the area. There is also a chain of 'in-use' TFCB's to reduce the search time for an existing table name. Each TFCB has a forward and backward chain pointer to facilitate rechaining. The base pointers to the 'in use' and 'free' TFCB's are in the first 64 bytes of the TFCB area (which starts with the tag TFCB), along with fields for Table Facility statistics gathering.

While a table is being built, the 31-Amode storage for it is under control of the caller's (application program) thread. Thus it is displayed in an indicative dump, and the address is given in the thread dump (see Appendix A). Once the table is closed, ownership of the table area is transfered to the system (Intercomm) so that the table is not purged and can be accessed later by another program. However, the program thread that has built the table can close it with a delete request and the table area is freed. If the program building the table should program check (terminate abnormally) or time out, the table is freed during program purge processing. If an application program has opened a table for update, its exclusive control of the table (via table name) is freed and the table is closed and kept by program purge processing if the program has not closed it. Shared control of a table is also freed and examined for continued validity on the next open request. This is accomplished via Intercomm internal enqueue (INTENQ and INTDEQ macros) processing.

2.3 PROGRAM INTERFACE

A user application program (and/or the Intercomm Page Facility) accesses the Table Facility via a CALL to the desired INTTABLE interface entry point. A brief description of the interfaces follows:

- TABUILD - build (create) a table with a unique (program-specified) name and giving the entry length, plus key length and offset in an entry (if desired).

- TABOPEN - open (for retrieval or update) an existing table, passing the table name.

- TABPUT - put an entry into a table via add of an entry at the end, or update/delete of an existing entry previously gotten for update.

- TABGET - get (sequentially retrieve) an existing table entry (optionally skip duplicates in a sorted table), or get the next or a specific entry (optionally for update or delete). A specific entry may be designated as the first, previous, next, or last, or first entry with a specific key, or a relative (to 1) entry number. Note that sequential retrieval may start at the end and be in reverse order, or may start after a specific entry is requested.

- TABSORT - sort a previously built or opened (for update) table containing keyed entries.

- TABEND - close a previously built (by the same program thread) or opened table and optionally keep or delete it.

While a program is building or accessing a table, it will have in its dynamic working storage (DWS) a copy of the Table Facility Control Block (its TFCBarea). This copy is initialized when the program calls TABUILD or TABOPEN and will contain a last retrieve 'pointer', the address of the table 'I/O' area in the program's DWS, the table entry area address, information about the table access being done by the program, and a 'pointer' to the system TFCB in storage above the 16M line (to speed access to the TFCB and table area by INTTABLE). If entries are being added to a new or existing table, the count of entries will be updated in both the system and program TFCB areas, as will the table entry area address (if changed) and the last entry 'pointer'. Thus, for debugging in a snap 126, the programmer can use the TFCB in the program's dynamic working storage to determine the current table status and activity. To easily find the TFCB area in the DWS, the area will begin with the table name (16 bytes) followed by the tag TFCB. Note that the pointers described above are actually fullword offsets into the associated area (system TFCB area or table area) from the address of the area.

This user TFCB area (TFUB) is one of the parameters passed on all Table Facility calls, along with a TFCW (Table Facility Control Word) for request options and return codes. The layout of the user TFCB area is described in Appendix A along with other debugging information.

2.4 TABLE USAGE SCENARIOS

Possible calling sequences for using the Table Facility are listed below:

a) Program calls TABUILD to create a table,
calls TABPUT for each entry to add to the table,
calls TABEND to keep table for future access.

b) Program calls TABUILD to create a table with keys,
calls TABPUT to add multiple entries to table,
calls TABSORT to sort created table,
calls TABEND to keep sorted table for future access.

c) Program calls TABUILD to create a table (with keys),
calls TABPUT to add multiple entries to table,
calls TABSORT to sort created table (optional),
calls TABGET to sequentially retrieve (sorted) entries,
calls TABEND to delete table.

d) Program calls TABUILD to create a table,
finds return code = 3 (table already exists),
calls TABOPEN to access table,
calls TABEND to delete existing table then,
follows scenario a, b, or c.

e) Program follows scenario a or b then,
calls COBPUT (MSGCOL) to pass a message with the table name to another program.

f) Other Program gets table name from passed message,
calls TABOPEN to access table,
calls TABPUT to add entries and/or,
calls TABGET to retrieve entries,
calls TABEND to keep or delete table.

g) Program calls TABOPEN to access an existing table,
calls TABPUT to add 1 or more entries at end
or
calls TABGET to get a (specific) entry for update
then
calls TABPUT to save updated (deleted) entry
or
calls TABGET to retrieve one or more entries without update/delete,
then
calls TABEND to close and keep table.

In all cases of creating (adding) or updating/deleting entries, the source of data for the entry (or delete request) may come from an operator terminal (operator keys data or delete request), or an existing file or Data Base record. The terminal request/data is passed to a program in the form of an input message. The message may contain a (generic) key of a data record or records to add/update/delete in a table or contain data to use as a basis for creating table entries from

existing data records. Data records must be retrieved by the application program using standard Intercomm interfaces. Conversely, an input message may request a program to retrieve (and format for transmission) a specific table entry, or a group of table entries from an existing table. Alternatively, an input message may request a program to create a table, then retrieve the entries (or retrieve entries from an existing table) for output formatting and printing on a system printer accessible by Intercomm.

In batch mode, the source of table entries may be program-generated or may be records from one or more files and/or Data Bases. Records may be retrieved, put in a table, sorted, and then printed or used to update a master file.

CHAPTER 3

USER PROGRAM INTERFACE

3.1 OVERVIEW

A user application program accesses the Table Facility via a CALL to the desired INTTABLE interface entry point. Parameters to be passed with the call are listed under each interface routine description and general return codes are also given. Note that while an Assembler Language or PL/1 program can call the interface routine entry point directly, an on-line COBOL program must call Intercomm's COBREENT (reentrant COBOL interface processor) passing as the first parameter the REENTSBS table routine name (provided in the COBOL copy member ICOMSBS) for the interface entry point, followed by the interface routine's parameters as given below. (See COBOL Programmer's Guide). All parameter areas (except the REENTSBS name) must be located in the dynamic working storage (save/work area, DSA or DWS) of the calling program. The addresses (labels) of these areas are passed on the CALL.

Note that each table entry is fixed in length - if an actual entry is smaller than the specified (at BUILD) length, low-order blank (spaces) or binary zero (low-values) padding must be provided by the caller. The first two bytes of the entry may contain the actual entry length (if variable) for application program usage. An entry may not have X'FF' (high-values) as the first 256 bytes (or as the entire entry if less than 256 bytes in length). X'FF' is used as a deleted entry indicator (entry may be inadvertently deleted).

3.2 PARAMETERS

For all calls, the first two parameters are the same and are:

- TFCBarea - 64 byte fullword-aligned area to contain the user program's copy (TFUB) of the Table Facility Control Block for the table being built or accessed. This area is initialized by the first call for a specific table and modified by all other calls for that table. If more than one table is concurrently accessed or built within a program, a unique TFCBarea must be supplied for each table. The layout of the TFUB is described in Appendix A. The TFUB may not be modified by a program after the initial call to TABUILD or TABOPEN. The TFUB may be reused after a call to TABEND.

- TFCW - a four byte (fullword) area to pass calling request options in byte 2 (in character form) and to receive a return code (in character form) in byte 1 (standard Intercomm application program Facility interface processing). For Assembler Language callers, the return code in hex (multiplied by 4) is also returned in register 15 (low order byte) and may be used for a branch table.

3.3 CALLING TF FUNCTIONS

For COBOL, the Table Facility service routines are invoked through calls via the interface routine COBREENT. The REENTSBS codes for TF functions are:

- TABUILD--36
- TABOPEN--40
- TABPUT---44
- TABGET---48
- TABSORT--52
- TABEND---56

The specific form of the CALL statement depends on the programming language being used. Examples are given below for all languages supported by Intercomm, where 'function' is the specific routine being called:

- Assembler Language:

  ```
  [symbol] CALL  function,(TFUB,TFCW[,other parms]),VL,MF=(E,list)
  ```

  where list is a parm-list area in the callers dynamic save/work area. For Assembler programs executing below the 16M line, the function address can be preloaded in register 15 from the SPAEXT (see Assembler Language Programmer's Guide).

- PL/1:

  ```
  CALL  function(TFUB,TFCW[,other parms]);
  ```

  where function is declared as ENTRY OPTIONS (ASM INTER); or is in copied member PLIENTRY

  or

  ```
  CALL  PMIPL1(function,TFUB,TFCW[,other parms]);
  ```

  where function is the REENTSBS-offset code label in copy member PENTRY (see PL/1 Programmer's Guide).

- COBOL:

  ```
  CALL  'COBREENT' USING function,TFUB,TFCW[,other parms].
  ```

  where function is the REENTSBS-offset code label in copy member ICOMSBS. For batch mode, use the actual entry point name instead of COBREENT and omit the function value.

In all cases, 'other parms' are defined as needed for the specific called entry point as described below.

3.3.1 Building a Table (TABUILD)

TABUILD is invoked to create (build) a new table with a unique (program-specified) name. At this time, the application program defines table characteristics such as its name, the entry length and optionally the key length and offset. A key length (may be the same as the entry length if 256 bytes or less) and offset must be given if the table will be sorted, even if specific access by key is not anticipated. The parameter list for TABUILD is:

```
TFUB,TFCW,table-name,entry-length[,key-length,key-offset]
```

where:

- TFUB (required) is the address of the 64-byte user TFCBarea to hold the copy of the new table TFCB initialized by TABUILD.

- TFCW (required) is the address of the 4-byte (fullword) user Table Facility Control Word where byte 2 contains:
  blank (or binary zeros/low-values) = no keys used, or
  K = each table entry will have a key.

- table-name (required) is the address of a 16-byte area containing the unique alphanumeric table-name (left justified with low-order blank padding if needed). This name will be placed in the first 16 bytes of the intialized TFUB area and therefore may be preplaced there for this parameter before the CALL. The name may not begin with a character zero (0) (reserved for Intercomm system usage).

- entry-length (required) is the address of a halfword (PIC S9999 COMP.) containing the (maximum) length to be used for every table entry (length of user program table entry 'I/O' area). The length can be from 1 to 32767, but may not be larger than the initial or increment table area sizes defined on the SPALIST macro for the Table Facility.

- key-length (required if option in TFCW = K) is the address of a halfword containing length of the area within each table entry which will contain the entry key. Key-length may be from 1 to 256 bytes and may be the same as the entry-length (if 256 bytes or less).

- key-offset (required if key-length given) is the address of a halfword containing the offset (relative to zero) to each key within each table entry: if the key is at the beginning of the entry, or is the same length as the entry length, then the offset is zero. The offset value may be from 0 to 255. The key must be completely contained within the first 256 bytes of the entry (for sort and search processing).

3.3.1.1 TABUILD Return Codes

The return code is in character in byte 1 of the TFCW (and in binary and multiplied by 4 in register 15 for Assembler Language callers). TABUILD return codes are:

R.C.	R15	Meaning
0	00	Request successful, TFCB areas initialized.
1	04	TFCW - invalid option request.
2	08	Table name not valid (not provided) or system not XA or ESA.
3	12	Table not built - table name exists.
4	16	No core to acquire/expand system TFCB area or to acquire initial table area.
5	20	Entry-length invalid or not provided.
6	24	Key-length provided but TFCW option not K.
7	28	Key-length missing but TFCW option is K.
8	32	Key-length invalid or greater than entry-length.
9	36	Key-offset invalid or missing.

NOTE: if return code is 3 - use TABOPEN instead (then TABEND with delete option if program desires to recreate table).

3.3.2 Opening a Table (TABOPEN)

TABOPEN is invoked to open (find) an existing table with the table name provided on the call, and to set up the user TFCB area (TFUB) for program access to the named table. A table may be opened for update to change, add or delete one or more entries and/or to sort the table (if keyed), or may be opened for (sequential) entry retrieval only. The parameter list for TABOPEN is:

```
TFUB,TFCW,table-name
```

where:

- TFUB (required) is the address of the 64-byte user TFCBarea to hold the copy of the system TFCB describing the requested table. The first 16 bytes may be initialized to table-name (if program has executed TABUILD, TABPUT's, TABEND (keep), and then wants to access those table entries, the same TFCBarea may be reused).

- TFCW (required) is the address of the 4-byte (fullword) user Table Facility Control Word where byte 2 contains:
  U = entries may be modified/added or table may be sorted (if needed), or
  R = entries will only be retrieved, or
  N = same as R, but skip duplicates if table sorted.

- table-name (required) is the address of the 16-byte area containing the name of the table to be accessed (may be in the first 16 bytes of the TFUB), which must be left-justified with low-order blank padding, if needed.

3.3.2.1 TABOPEN Return codes

The return code is in character in byte 1 of the TFCW (and in binary and multiplied by 4 in register 15 for Assembler Language callers). TABOPEN return codes are:

R.C.	R15	Meaning
0	00	Table exists and is available for access, user TFUB area initialized.
1	04	TFCW - invalid option request.
2	08	Table name not provided or invalid parameter list.
3	12	Table name found, but table is already being updated/built by calling program.
4	16	Table name found, but table is still being built by another program (thread) and has not yet been closed for later access.
5	20	Table name found, but table is being accessed for update by another program which has exclusive control of the table.
6	24	Table name found, but table is being accessed by other program(s) (not available) when caller's TFCW option is U.
7	28	Table name found, but table has no data entries. Entries may be added if opened with TFCW option U, otherwise, TABEND (keep) the table and reopen it with option U or TABEND (delete) the table and rebuild it.
8	32	Table name found, but an ENQ is in effect for the name which was not issued by INTTABLE, or table access count is at maximum (255 threads). Table cannot be accessed.
9	36	Table name not found.

NOTES:

- return codes 1, 2, 3, or 8 = programming error;
- return codes 4, 5, or 6 = program may call Intercomm's IJKDELAY to temporarily give up control (allow other program(s) to finish table access) before retrying TABOPEN call;
- return code 0 = byte 2 will contain a character:
  X - if table has deleted entries (not sorted), or
  S - if table is sorted (no duplicates), or
  D - if it is sorted and has duplicates, or
  the passed option code if none of the above applies;
  bytes 3 and 4 will contain the length of a table entry (program's table entry 'I/O' area length must be at least as large as this length).

3.3.3 Placing an Entry in a Table (TABPUT)

TABPUT is invoked to place an entry in a table via add of an entry to the end of a table (existing or being built) or via update/delete of an existing entry previously retrieved (via TABGET) for update; add requires a previous TABUILD or a previous TABOPEN with option U, update/delete requires a previous TABOPEN with option U. The parameter list for TABPUT is:

```
TFUB,TFCW,entry-area
```

where:

- TFUB (required) is the address of the 64-byte user TFCBarea as initialized by a call to TABUILD or TABOPEN for the same table. This area may not be modified by the user program between calls.

- TFCW (required) is the address of the 4-byte (fullword) user Table Facility Control Word where byte 2 contains:
  A = add entry at end of table, or
  U = update previously retrieved entry, or
  D = delete previously retrieved entry.

- entry-area is the address of the user area containing the entry to be added/updated/deleted. This parameter may be omitted if option=D: in the table data area the entry will be set to X'FF' (high-values) and cannot later be accessed (automatically skipped if sequential retrieval), except by relative entry number (see TABGET). If the deleted entry is in a sorted table, it will automatically be removed by TABPUT if it is the first or last table entry, otherwise the table is flagged for resort.

3.3.3.1 TABPUT Return Codes

The return code is in character in byte 1 of the TFCW (and in binary and multiplied by 4 in register 15 for Assembler Language callers). TABPUT return codes are:

R.C.	R15	Meaning
0	00	Entry added/updated/deleted.
1	04	TFCW - invalid option request.
2	08	No previous TABUILD or TABOPEN (invalid TFCBarea), or invalid parameter list.
3	12	Entry-area not passed for option A or U.
4	16	No storage to add another entry.
5	20	Request = U or D but entry not previously retrieved for update.
6	24	Request = A/U/D but TABOPEN not for update/add.
7	28	Request = U or D but table still in BUILD status.
8	32	All table entries deleted (program should call TABEND with delete, or create (add) new entries).

3.3.4 Retrieving an Entry from a Table (TABGET)

TABGET is invoked to get (sequentially retrieve) an existing table entry (optionally skip duplicates if table sorted) or to get the next/specific entry (optionally for update or delete). A specific entry may be designated as first, previous, next, last, or the first entry with a specific key, or a relative entry number (relative to 1) may be requested. The parameter list for TABGET is:

```
TFUB,TFCW,entry-area[,{key-area    }]
                      {entry-number}
```

where:

- TFUB (required) is the address of the 64-byte user TFCBarea as initialized by a call to TABUILD or TABOPEN for the same table. This area may not be modified by the user program between calls.

- TFCW (required) is the address of the 4-byte (fullword) user Table Facility Control Word where the following options may be used depending on the desired retrieval type:
  byte 2:
  - R = retrieve first/next entry even if next entry has a duplicate key (skip next entry if duplicate key in sorted table opened with option N).
  - N = retrieve next entry (skip next entry if duplicate key in sorted table and retrieve next non-duplicate entry).
  - U = retrieve next/specific entry for update or delete.
  - S = retrieve specific entry (no update/delete).

  byte 3 (if option in byte 2 is U or S):
  - K = retrieve specific entry with provided key (first entry with key if duplicates) - presorted table only.
  - R = retrieve specific entry with provided entry number (even if entry flagged for deletion).
  - F = retrieve first valid entry.
  - L = retrieve last valid entry (even if duplicate).
  - P = retrieve previous entry (even if duplicate key, unless table opened with option N).
  - N = retrieve next entry (default) even if duplicate key, unless table opened with option N.

  NOTES: the user TFUB contains a 'pointer' to the last retrieved entry, which is used as a base for next or previous retrieval. To start or restart retrieval at a specific point, use option S (or U) in byte 2 and the desired starting point option (except N) in byte 3. To then continue previous entry retrieval, option SP (or UP) in bytes 2 and 3 must be used. However, to then continue next entry retrieval, option R or N (in byte 2) or UN (in bytes 2 and 3) may be used.

If option R or N is used (in byte 2), INTTABLE places an N in byte 3 for internal processing (may be ignored by the program). However, if no previous access to the table has been made by the program since a TABSORT or TABOPEN for the table, INTTABLE places an F in byte 3.

If a program has built a table and put entries in it and then desires to retrieve the entries, option SF or SL must be used on the first TABGET call to initiate retrieval from the beginning or end of the table. Otherwise a return code of 6 for the TABGET call will result if option R or N used in byte 2 (trying to retrieve beyond table end).

Duplicate entries are tested for only in presorted tables.

Once a keyed entry has been updated with a key change, then the table cannot be accessed by key until it has been resorted (see Return Code 1): either call TABSORT before the next keyed access, or call TABEND (automatically resorted), then reopen the table. Sequential or entry-number access may be used for other entries without resorting.

- entry-area (required) is the address of the user 'I/O' area to contain the retrieved entry (if found/available). The size of this area must be of the maximum entry length defined when the table was built (see NOTES under TABOPEN return codes).

- key-area is the address of the user area containing the key of the specific entry to be retrieved from a sorted table if option SK or UK was used.

- entry-number is the address of a fullword containing the number (relative to 1) of the entry to be retrieved if option SR or UR is used. For example, to retrieve the sixth entry in the table, place a 6 in the fullword (binary value).

3.3.4.1 TABGET Return Codes

The return code is in character in byte 1 of the TFCW (and in binary and multiplied by 4 in register 15 for Assembler Language callers). TABGET return codes are:

R.C.	R15	Meaning
0	00	Requested/next entry successfully retrieved (in user entry-area).
1	04	TFCW - invalid option request or request combination, or K request in TFCW byte 3 invalid (no keys or table not sorted or needs resorting).
2	08	No previous TABUILD or TABOPEN (invalid TFCBarea), or invalid parameter list vs. options.
3	12	No entry-area provided.
4	16	Option byte 2 = U or S and option byte 3 = K or R but no key area or entry-number provided.
5	20	Request = U but TABOPEN not for update or still in TABUILD mode.
6	24	Entry not found: relative number invalid or beyond table end (option SR or UR); or key not found (option SK or UK); or beyond end of table for 'next' entry request.
7	28	Entry not found, request was P (previous), but last GET retrieved first valid entry.
8	32	Entry not found, request was L (last), but previous GET retrieved last valid entry.
9	36	Table has no entries (all deleted).

NOTE: Byte 4 will be used for special information codes when the return code is 0, as follows:

D = option byte 2 = R, key of entry returned from sorted table is a duplicate of previous entry.

X = option byte 2 = U or S and option byte 3 = R but specific entry returned is a previously deleted entry (contains high values).

R = same entry as previously requested (preceding TABGET call) returned if option byte 2 = U or S and byte 3 = K or R.

L = last valid entry in table returned.

F = first valid entry in table returned.

The default is a blank if none of the above applies (SPACE or X'40').

3.3.5 Sorting a Keyed Table (TABSORT)

TABSORT is invoked to sort a previously built or opened (by the program) table based on the user-defined (at BUILD) key area in the table entries. A previously opened table should have been opened with option U (for exclusive control). If it was not, TABSORT attempts to acquire exclusive control, but if it cannot, then the request is rejected (see TABSORT Return Codes). The parameter list for TABSORT is:

```
TFUB,TFCW
```

where:

- TFUB (required) is the address of the 64-byte user TFCBarea as initialized by a call to TABUILD or TABOPEN for the table to be sorted.
- TFCW (required) is the address of the 4-byte (fullword) user Table Facility Control Word. No options are used.

3.3.5.1 TABSORT Return Codes

The return code is in character in byte 1 of the TFCW (and in binary and multiplied by 4 in register 15 for Assembler Language callers). TABSORT return codes are:

R.C.	R15	Meaning
0	00	Table sorted.
1	04	(TFCW - not applicable).
2	08	Invalid TFCBarea (table not previously built or successfully opened) or invalid parameter list.
3	12	No key (length/offset) defined when table originally built.
4	16	Table not available for sort - other program(s) accessing table (table not opened with option U).
5	20	Table has no entries (nothing to sort or all deleted).

NOTE: Byte 2 of the TFCW will contain a character D if duplicates found while sorting.

3.3.6 Closing a Table (TABEND)

TABEND is invoked to close a table previously built or opened and to specify whether to keep it or delete it - keep implies that the table will be accessed subsequently by the same or other program(s), while delete requests system TFCB clean up and freeing of table entry area storage; in either case, the program's TFCBarea flags will reflect the closed status of the table (see Appendix A). The parameter list for TABEND is:

```
TFUB,TFCW
```

where:

- TFUB (required) is the address of the 64-byte user TFCBarea as initialized by a previous call to TABUILD or TABOPEN for the table to be closed.

- TFCW (required) is the address of the 4-byte (fullword) user Table Facility Control Word where byte 2 contains:
  K = keep table for later access, or
  D = delete table and clean up control blocks.

3.3.6.1 TABEND Return Codes

The return code is in character in byte 1 of the TFCW (and in binary and multiplied by 4 in register 15 for Assembler Language callers). TABEND return codes are:

R.C.	R15	Meaning
0	00	Table closed as requested.
1	04	TFCW - invalid option request.
2	08	Invalid TFCBarea or invalid parameter list.
3	12	Table already closed (previous TABEND for table name requested).
4	16	Table cannot be deleted because other program(s) still accessing it.

NOTE: if option is K and the table has keys and is flagged for resort (due to entry add or delete, or key change update by caller), then the table is automatically resorted by TABEND processing, and the flag is turned off. Deleted entries will also be removed if the table is resorted (see TABPUT).

Chapter 4

INSTALLATION AND SNAPS

4.1 TABLE FACILITY INSTALLATION

Installation of the Table Facility requires only two steps, as follows:

- Linkedit of INTTABLE with other system programs.
- Definition of Table Facility SPALIST parameters.

There are no files or JCL statements used by the Table Facility.

4.1.1 Intercomm Linkedit

Ensure that INTTABLE is included in the resident portion of the Intercomm load module. Also ensure that the following system modules used by INTTABLE are included: PMINQDEQ, INTSORT and BINSRCH (for sorted tables with keys), RMPURGE (program cleanup/purge processing) and TDUMP (Thread Resource Dump processing), FDITCB (application thread control block processing - examined by RMPURGE), and STRTSTOP (disable Table Facility snaps at startup - see Section 4.2).

Also ensure that the version of MANAGER included in the linkedit has been assembled with the &RM global in SETGLOBE set to 1, to generate RCB table processing to track INTTABLE acquired storage areas and to ensure correct processing of STORAGE, STORFREE, and PASS macro requests for 31-Amode core.

Note that INTTABLE, INTSORT and BINSRCH are serially reusable (use a local save area), and that INTSORT and BINSRCH may be called in either 24-Amode or 31-Amode to process tables located below or above the 16M line.

4.1.2 Subsystem SYCTTBL Definition Considerations

The TCTV timeout value should be generous for subsystems which use file or Data Base I/O to build a table, or which request a table sort either directly (calls TABSORT) or indirectly (table resorted at TABEND with keep after an entry changed or added).

4.1.3 SPALIST Parameters

The following parameters on the SPALIST macro (which generates the SPA and SPAEXT Csects) may be specified in the SPA member included in the Intercomm region linkedit where the Table Facility is used. Initially, the defaults may be used and then modified as Table Facility usage increases. Statistics for tuning these parameters are described in Chapter 5. These parameters may not be dynamically modified while Intercomm is executing. The SPALIST macro parameters are:

- TFCBINT - define the size (in K) of the initial system area to hold Table Facility TFCBs (which are each 64 bytes in length), and the TFCB control area (first 64 bytes). Each 1K will hold 16 TFCBs. The default is 4 (4K or 4096 bytes which holds 63 TFCBs plus the control area).

- TFCBADD - define the size in K of the TFCB area increment to acquire to hold additional TFCBs. Note that a TFCB area is reused after a table is deleted. Expansion of the TFCB area is called a TFCB area relocation. A relocation count statistic is kept and may be used to ultimately define the optimum size for TFCBINT to eliminate relocation processing overhead. The default for TFCBADD is 4 (4K).

- TABINT - define the size in K of the initial table entry area to acquire for each table when it is built (created). The minimum must be large enough to hold the largest table entry to be created. The default is 4 (4K or 4096 bytes).

- TABADD - define the size in K of the table area to acquire to expand a table entry area for a table (if needed to add more entries). The minimum and default are the same as for TABINT.

- TABSP - specify the MVS subpool number, from 1 to 127, to be used to acquire all Table Facility 31-Amode storage (above the 16M line). The default is subpool 0.

After adding/changing the SPALIST parameters, reassemble the SPA member and relinkedit it with the Intercomm load module.

4.2 TABLE FACILITY ENTRY PROCESSING SNAPS

Table Facility snaps of entry processing (calls to TABPUT and/or TABGET) are available for testing table access processing. Under Intercomm, table snaps are deactivated at startup and are dynamically controlled via the STRT and STOP commands. The parameter option is TABSNAP. To start system-wide snaps of table processing, use the following command:

```
STRT$TABSNAP
```

Henceforth, all calls to TABPUT or TABGET by all programs (system and user) using the Table Facility will result in individual snaps being written to the SNAPDD data set. Table entry processing snaps may subsequently be deactivated via the following command:

```
STOP$TABSNAP
```

Snap-ids are 130 for TABGET snaps and 131 for TABPUT snaps.

Areas snapped are:

- SAVE AREA TRACE - calling sequence trace

- TF Snap Control Block giving the thread's processing request input message terminal-id (TID), the processing subsystem's codes (SS CODE), the Table processing MODE (BUILD or OPEN), the Table Facility entry CALL (GET or PUT) and the TFCW option(s) passed for the call. For example:

```
 TID   SS CODE  MODE   CALL TFCW
TEST2 E3C6.TF  OPEN   GET   SR
```

  This area will always be boundary-aligned in the snap for searching on TID or SS CODE through a group of snaps.

- System TFCB - 64-byte Table Facility Control Block for table being accessed, as modified (if needed) after processing the call. The table name is in the (aligned) first 16 bytes.

- User TFCB - user 64-byte TFUB area passed for the call after modification for processing the call. The table name is in the first 16 bytes, followed by the 'TFCB' identifier.

- User Entry Area - containing the entry retrieved from the table, or put (updated/added) into the table. This area is omitted if not passed for a delete request on a TABPUT call.

The layout of the TFCB and TFUB is described in Appendix A. If the call was to PUT for a delete, the snap is taken before delete processed.

To use Table Facility snaps, ensure STRTSTOP and PMISNAP1 are included in the Intercomm linkedit, and that the SNAPDD data set is defined (see Operating Reference Manual).

See Chapter 6 for batch mode processing with Table Facility snaps.

Chapter 5

TABLE FACILITY STATISTICS

5.1 STATISTICS OVERVIEW

Statistics on Table Facility processing are gathered by INTTABLE and stored in the TFCB control area (first 64 bytes) of the TFCB area. Statistics include total tables built, current number of tables (TFCBs in use), current and maximum table entry area space acquired, total table area expansions and TFCB area relocations, average table size, the largest single table area size and the maximum entries created in one table. These statistics are provided (printed) in the System Tuning Statistics.

Additionally, Core Use Statistics processing tracks 31-Amode storage requests and usage. The user may track Table Facility calls via SAM (System Accounting and Measurement) processing. Some statistics may be dynamically displayed via the TALY$SU command, and Table Facility control blocks and table areas may be dynamically displayed via the SCTL command. Thread dumps also list storage acquired by INTTABLE and give the subpool number and full 31-Amode address (see Appendix A).

5.2 CORE USE STATISTICS

Under the general heading Core Use Statistics, in addition to statistics for 24-Amode core requests, statistics are reported for 31-Amode requests (via LOC=ANY parameter on STORAGE macro). For the line starting '31-AMODE CORE:', the STORAGES ISSUED value is cumulative to the point of printing the statistics (since system startup). The DOUBLE WORDS NOW IN USE is the current value while HIGH THIS RUN (on the same line) is the maximum concurrent doublewords in use since system startup. Note that these statistics may include system and/or user requests for 31-Amode storage other than via the Table Facility.

See the Operating Reference Manual for further details on Core Use Statistics which are periodically printed by RMTRACE on the SMLOG (SYSOUT) data set.

5.3 SYSTEM TUNING STATISTICS

After the first call to the Table Facility, the system tuning statistics module INTSTS (if included in the linkedit) will produce Table Facility processing statistics on its STSLOG (SYSOUT) data set (if defined in the execution JCL - see Operating Reference Manual). Under the TABLE FACILITY STATISTICS heading, the following data is given:

For the TFCB area:

- TFCBINT - value coded for TFCBINT on the SPALIST macro.
- TFCBADD - value coded for TFCBADD on the SPALIST macro.
- CURRENT AREA - current size of the TFCB area.
- RELOCATIONS - cumulative total TFCB area relocations.
- # TABLES - current number of defined tables (includes tables being built, but not yet closed; does not include previously deleted tables).

For the Table Entry Areas:

- TABINT - value coded for TABINT on the SPALIST macro.
- TABADD - value coded for TABADD on the SPALIST macro.
- CURRENT AREA - current total space acquired for table entry areas.
- EXPANSIONS - total expansions of table areas (cumulative).
- # BUILT - total created tables (cumulative).
- MAX TABLES - maximum concurrent defined tables at any time during this system execution.
- MAX AREA - maximum concurrent table area space acquired at any time during this system execution.
- AVERAGE SIZE - current average table size (CURRENT AREA divided by # TABLES).
- MAX ENTRIES - maximum entries created in one table at any time during this system execution (checked when table closed (TABEND called), even if deleted).
- MAX TAB SZ - largest table created at any time during this system execution (checked when table closed (TABEND called), even if only built, then deleted (not kept)).

Note that the AVERAGE SIZE statistic may not be realistic for user tables if the Intercomm Page Facility (or other system facility) is also using the Table Facility.

To reduce the number of RELOCATIONS, increase the TFCBINT value. To reduce the number of EXPANSIONS, increase the TABINT value.

5.4 SAM TRACKING OF TABLE FACILITY CALLS

Under the System Accounting and Measurement Facility (SAM) for tracking on-line application program (subsystem) activity and resource usage, the following tracking parameters ('buckets') may be specified via the MAPACCT macro in the user-coded SAMTABLE:

- TABUILDS - accumulate count of calls to TABUILD
- TABOPENS - accumulate count of calls to TABOPEN
- TABPUTS - accumulate count of calls to TABPUT
- TABGETS - accumulate count of calls to TABGET
- TABSORTS - accumulate count of calls to TABSORT
- TABENDS - accumulate count of calls to TABEND.

SAM statistics are gathered for each subsystem for which SAM=YES (default) is defined on its SYCTTBL macro on a thread by thread (message) basis. Bucket accumulators may be combined. Based on SAM statistics for specific sybsystems using the Table Facility, the number of calls made vs. the expected number for the subsystem processing path may be checked for processing efficiency, or for system resource usage activity and accounting. Note that reported Table Facility calls may include system usage of the Table Facility, such as via the Page Facility. See the Operating Reference Manual for SAM usage, installation and reporting.

5.5 ONLINE TABLE FACILITY PROCESSING DISPLAYS

The TALY$SU command display (see System Control Commands) gives the following information on Table Facility processing:

- TOTAL TABLES - current number of defined tables (same as # TABLES on STS statistics report).
- TABLE CORE IN USE - current total space acquired for table entry areas.
- TFCB RELOCATIONS - cumulative total TFCB area relocations (same as RELOCATIONS on STS statistics report) - see Chapter 4 (SPALIST macro TFCBINT and TFCBADD parameters).

The SCTL command (see System Control Commands) can be used to dynamically display or print Thread Resource Dumps of system or user resources (see Appendix A), or to find the address of the system TFCB area, or to display or print (as snaps) the system TFCB area or a user table. See Appendix A for locating the addresses of these areas.

Chapter 6

BATCH MODE PROCESSING

6.1 USING THE TABLE FACILITY IN BATCH MODE

In batch mode, the Table Facility can be used as a scratch pad area, or to store condensed versions of data records to produce batch reports, or to store and then sort update records to process against a file or Data Base, for example.

The Table Facility may be accessed by batch user application programs if the following linkedit is used:

```
INCLUDE SYSLIB(user-program)   user application program
INCLUDE SYSLIB(INTTABLE)       Table Facility program
INCLUDE SYSLIB(PMINQDEQ)       INTENQ/DEQ program
INCLUDE SYSLIB(BATCHPAK)       Intercomm psuedo entry points
```

Note that BATCHPAK has entry points to process common Intercomm macro requests including STORAGE and STORFREE requests. The STORAGE entry point also supports 31-Amode storage requests (LOC=ANY parameter) and the STORFREE entry point will free 24-Amode and 31-Amode storage, depending on area address. Therefore, INTTABLE in batch mode (when linked with BATCHPAK) will acquire TFCB and table area core in 31-Amode as in on-line processing. PMINQDEQ must be included before BATCHPAK.

BATCHPAK also contains SPA and SPAEXT Csects which contain the default values for TFCBINT, TFCBADD, TABINT, TABADD and TABSP (see Chapter 4). If a larger value than the default (4K) for TABADD and TABINT is desired, the on-line SPA module (or a specially coded version) should be included before BATCHPAK (to override the Csects in BATCHPAK). Coding of TABSP as other than 0 (default) is ignored (not supported by BATCHPAK).

If the created table(s) will have keys and sorting of a table and access by key is desired, also include INTSORT and BINSRCH in the linkedit (before BATCHPAK).

If snaps (see Chapter 4) of table access activity is desired, include PMISNAP1 in the linkedit, and define the SNAPDD data set for snap output in the JCL. Also, it will be necessary to OPEN the snap data set for output. The following Assembler Language statements are needed:

```
EXTERN PMISNAP
OPEN (PMISNAP,(OUTPUT))
```

If the user application program is written in COBOL or PL/1, it will be necessary to code an Assembler 'top hat' program to open the snap data set (include before the user application program and call the user program from it - do not forget to chain save areas). If the user table processing program is written in Assembler Language, production of snaps may be dynamically controlled via the SSSTART and SSSTOP macros with TYPE=TABSNAP (see Basic System Macros).

Other Intercomm on-line facilities that may be used in batch mode include:

File Handler - see Operating Reference Manual
Store/Fetch - see Store/Fetch Facility
DDQ - see Dynamic Data Queuing Facility

Of course, standard data set and/or Data Base access may be used (see vendor manuals).

Appendix A

DEBUGGING TABLE ACCESS PROBLEMS

A.1 INTRODUCTION

To validate table entry access, snaps may be used as described in Chapter 4. If a program check or timeout occurs in an application program using the Table Facility, then table usage information depends on whether the application is building a table or has opened it for retrieval and/or update. That is, if a program is building a table, the table entry area storage belongs to the program thread and will be snapped as a resource owned by the thread in an indicative dump (note that it will have a 4-byte or 31-Amode address). Whereas, if the program has opened an existing table for access, that table's storage belongs to the system and will not be snapped in an indicative dump. In a full region dump, all table areas and the TFCB area will be snapped in the subpool storage area near the back of the dump. Because they are in 31-Amode core (31-bit addresses), they should be easy to locate. The TFCB area is identified with the name TFCB in the first 4 bytes.

A.2 USING THE THREAD RESOURCE DUMP

Thread Resource Dumps are produced in the on-line system to list resources owned by all currently assigned thread numbers (for message processing) and for the system thread. They are produced after a program check or timeout, after a system abend, or if an application thread has not freed all resources acquired by the thread (such as closing a table) before returning to the system. Resources (CORE, FILE, NQ, etc.) are listed in reverse order of acquisition, that is, the most recently acquired resource is listed first. Under resources listed for the system thread (thread 0), CORE acquired by INTTABLE describes table areas, except that one of the areas (usually the first acquired) is the TFCB area. If a very small (less than 1024 bytes) area is listed as acquired by INTTABLE, it will be the saved (for reuse) snap header area (if table snaps were activated).

The thread that had a program check, timeout, or unfreed resources, is the owner of the FILE resource SMLOG (SYSOUT data set to which the thread dump is written), as its first resource. CORE resources owned by INTTABLE under the application thread are for table areas of tables being built by the thread - one for each table. There will also be an NQ resource for every table (name on the right) being accessed by the thread for which TABEND has not been called. If it is NQ(OWNER), then the thread has exclusive control of the table, and is either building the table or has opened it for update. If it is NQ(SHARE), then the thread has opened the table for retrieval only.

An enqueue is issued on the table name when TABUILD or TABOPEN is called and a dequeue is issued when TABEND is called (the NQ resource is freed). There is no enqueue timeout. Therefore, if more than one program (thread) needs access to a table, the second may time out waiting to access the table (see TABOPEN return codes). It is not put

in an NQ(WAIT) state by INTTABLE, but may put itself into a timed delay loop waiting to access the table. Check for NQ resources for tables owned by other threads to resolve this problem. Then determine why the owner thread has tied up the table - perhaps TABEND should be called as soon as the program has finished with the table, rather than just before returning to the system.

A.3 USING THE TFCB AREA

As stated above, the TFCB area containing Table Facility Control Blocks is easily identified in 31-Amode storage by the literal TFCB in its first 4 bytes. The address of the area is placed in the SPAEXT in the field labeled SEXTFCBP at offset X'16C' (see the SPAEXT Dsect). The first 64 bytes of the area is a header area for statistics gathering. The second and third fullwords of the header contain the offsets (from the TFCB area address) to the free TFCB chain and to the 'in use' (for existing tables) TFCB chain, respectively. All TFCBs in the area are on one or the other chain, the last in a chain contains a forward pointer of zero (binary zeros). All TFCBs within a chain are forward and back chained. Each chain is a 'push-down' stack.

Each TFCB on the 'in use' chain starts with the 16-byte table name. The next 2 fullwords in every TFCB contain the back and forward chain offsets (from the TFCB area address to the next TFCB) respectively. The TFCB with a back chain of zero is the first on the chain. A TFCB on a free chain may have a table name from an earlier usage or may have zeros in the name area (never used). If both chain offsets are zero in a TFCB, that TFCB is the only one on the chain. The next word following the 2 offset fullwords is the address of the table area for the named table. If the address is zero but there is a table name, then the TFCB is probably on the free chain (TABEND called to delete the table and free the TFCB). However, the address could be zero on an 'in use' TFCB if entries were placed in the table, then all entries were subsequently deleted (causing the table area to be freed). In the latter case, the next 2 fullwords, which normally contain the length of the table area and the number of entries in the table, would also be zero. These 2 words are not zeroed when a deleted table is placed on the free chain. Other fields in the TFCB are described in the TFUB/TFCB layout given in the next section (Figure A-1). Note that there are 2 flag bytes for table access status, and that one of the flags indicates if a TFCB is on the free chain. If the flag is on, this verifies the free status of the TFCB (see Figure A-2).

Whenever a new table is to be built, or an existing table is opened for shared or update access, the TFCB for the table is placed at the head of the 'in use' chain (rechained) by TABUILD/TABOPEN processing. The physical TFCB is not moved. Thus, in a small TFCB area, a TFCB for a specific table name (if alphanumeric names used) may be easily found by scanning the literals on the right side of the dump. In a large TFCB area, a table being currently accessed may be found by down-chaining (adding each forward offset to the TFCB area address) along the 'in use' chain from the TFCB header area.

A.4 USING THE USER TFCB (TFUB) AREA

The user TFUB is initialized when TABUILD or TABOPEN is called. Most fields in the TFUB have the same values (usage) as in the system TFCB except for several fields unique to user processing. For example, instead of a chain field, the first word following the table name contains the identifier literal TFCB. Thus a TFUB is easily found in the users dynamic working storage area. The second word following the 16-byte name area contains the address of the user's table entry 'I/O' area in the user's dynamic working storage passed on the most recent call to TABPUT or TABGET or binary zeros if not yet called. The third word following the name also contains the address of the 31-Amode table entry area as in the TFCB. Use this address to find the table entry area in a full dump or to find it in an indicative dump if the table is being built, and to validate the data in the named table.

Note that if a program is accessing more than one table, a unique TFUB area for each table must be used, and that the same area must be used on each CALL for the associated table name. INTTABLE validates that the TFUB is legitimate for a call to TABPUT or TABGET, but cannot tell if it is the TFUB for another table also being accessed by the same thread before moving an entry between the user entry area and a table. It will use the entry length stored in the associated TFCB and may move too much (causing a storage overlay and potential program check), or too little (causing an invalid entry), if the wrong TFUB is used for the CALL.

The layout of the 64-byte user TFUB and system TFCB is given in Figure A-1, and of the flag byte values in Figure A-2.

The system COPY member INTTABDS contains the Dsects describing the TFCB header, TFCB and TFUB.

All fields in the TFUB and TFCB except the table name contain hexadecimal values. The table name is a character field, but may contain hex values depending on how the user constructed the name.

TFUB Name	Length	Description	Description	Length	TFCB Name
NAME	16	Table name	Table name	16	NAME
TAG	4	'TFCB' literal	Back chain offset/ 0 if first	4	PREV
REC	4	User DWS entry area address/0	Forward chain offset/ 0 if last	4	NEXT
DATA	4	Table area address/0	Table area address/ 0 if freed	4	DATA
DTSZ	4	Table area length/0	Table area length/0	4	DTSZ
ENUM	4	Total table entries/ 0 if no entries	Total table entries/0	4	ENUM
ELEN	2	Entry length	Entry length	2	ELEN
KLEN	1	Key length minus 1/0	Key length minus 1/0	1	KLEN
KOFF	1	Offset to key in entry/0	Offset to key in entry/0	1	KOFF
READ	4	Offset to last entry retrieved from table/0	(reserved)	4	READ
ENDE	4	Offset to last valid table entry (0 if 1 or no entries)	Offset to last valid entry	4	ENDE
TFCB	4	Offset to associated system TFCB in TFCB area	Address of associated TFUB if in build or update mode	4	DWS
UKEY	6	Caller's unique key for purge processing	Unique key of table builder or last open for update	6	UKEY
FLG1	1	Flag Byte 1	Flag Byte 1	1	FLG1
FLG2	1	Flag Byte 2	Flag Byte 2	1	FLG2
	4	(reserved)	(reserved)	4	-

Figure A-1 TFUB/TFCB Area Layout

The flag bytes have the following settings if the associated bit is on (set to binary 1):

FLG1	
X'80'	- Table closed (with keep) by builder.
X'40'	- Table is in build mode (being built).
X'20'	- Table has been opened. (If on in the TFCB, but not in the TFUB, TABOPEN was for shared access.)
X'10'	- Table opened for update (exclusive control).
X'08'	- Table has been sorted.
X'04'	- Sorted table has duplicate keys.
X'02'	- Sorted table needs resorting.
X'01'	- Table entries have keys (if KLEN is 0, then the actual key length is 1).

FLG2	
X'80'	- Middle entry in table deleted (skip on retrieval, delete after resort).
X'40'	- TFCB is for a Page Facility terminal.
X'20'	- Page Facility Control Table TFCB.
X'10'	- Enqueue in effect for table name.
X'08'	- TFCB was used, now on free chain.
X'04'	- (reserved).
X'02'	- TFUB only - Table opened with option N - automatically skip duplicates in keyed table.
X'01'	- TFUB only - last call to TABGET was for update (set off on next call to TABGET/PUT/SORT).

Figure A-2 TFUB/TFCB Flag Indicators

Appendix B

INTENTIONAL PROGRAM CHECKS

B.1 DESCRIPTION

If an unrecoverable error (should not occur) is encountered by INTTABLE, it will force a SOC1 program check via the ISK (hex 09) instruction. The ISK operands used are given in the on-line PROGRAM CHECK message (MP001I) - see Messages and Codes. In a dump, the ISK code 2,0, for example, would be printed as hex 0920. The ISK codes and reason description are listed below. Furthur clarification is provided in Appendix A.

CODE	CAUSE
2,0	TABOPEN called with Page Facility Master Table name, but its TFCB offset stored in SPAEXT (in field SEXPFCBO) does not point to correct TFCB. Note that the offset is divided by 64 before being stored.
2,1	TABOPEN called with table name found on 'in use' chain, but free chain flag is on.
2,2	TABOPEN called and no NQ is in effect for the table name, yet the TFCB flag settings show the table being accessed in build or update mode (requires exclusive NQ). Possibly caused by RMPURGE not in linkedit, or correct version (to free tables) not in linkedit.
2,3	TABOPEN called and correct TFCB found, but it contains invalid data: offset to last valid table entry greater than number of entries in table.
2,4	TABOPEN called and correct TFCB found, but it contains invalid data: number of entries in table greater than offset to last valid entry.
3,0	TABPUT called and processing mode of table is build or update, however, NQ in effect for table is not for exclusive control.
4,0	TABGET called and processing mode of table is build or update, however, NQ in effect for table is for shared control (not allowed).
5,0	TABGET, TABPUT or TABSORT called and user TFUB is valid, however no NQ for the table was issued by INTTABLE, or it has been incorrectly cancelled.
5,1	TABGET, TABPUT or TABSORT called and user TFUB is valid, however NQ issued by INTTABLE no longer in effect for the table: cannot be accessed by caller.

INDEX

INDEX

INDEX