UNISYS

System 80 OS/3 Transaction Platform System (TPS)

Programming Guide

Volume I

IMPORTANT NOTE!

This documentation describes the *Transaction Platform System* (TPS) software product which is a fully functional subset of Allinson-Ross Corporation *Transaction Interface Processor /30* (TIP/30).

All references to TIP/30 in this documentation can be understood to refer to TPS.

OS/3 Release 13

August 1990

Printed in U S America

Priced Item

7002 3981-100

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THIS DOCUMENT. Any product and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a duly executed Program Product License or Agreement to purchase or lease equipment. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such License or Agreement. Unisys cannot accept financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, indirect, special, or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Correspondence regarding this publication should be forwarded to Unisys Corporation either by using the Business Reply Mail form at the back of this manual or by addressing remarks directly to Unisys Corporation, OS/3 Systems Product Information Development, P.O. Box 500, Mail Station E5-114, Blue Bell, Pennsylvania, 19424, U.S.A.

About This Document

Purpose

The *TIP/30 Programming Reference Manual Volume I*:

- introduces you to the TIP/30 product
- illustrates some fundamental concepts and facilities of TIP/30 and
- describes, in detail, the numerous TIP/30 utilities.

Scope

This document provides detailed descriptions of the numerous TIP/30 utilities, alphabetically ordered to facilitate ease of use.

Audience

The primary audience for this document is data processing programming staff.

Prerequisites

Anyone using this document should be familiar with Unisys computer hardware and the OS/3 software system.

In addition, knowledge of the COBOL programming language is an asset.

How to Use This Document

This document contains documentation for all of the TIP/30 utility transaction programs, arranged in alphabetic order. You should read this entire document to familiarize yourself with its contents and to determine which of the TIP/30 utilities are useful for your programming application.

Organization

This document contains five sections:

"Section 1 Overview of TIP/30"

This section provides a general description of the capabilities of TIP/30.

"Section 2 Fundamental Concepts"

This section describes fundamental concepts that are essential to the understanding of the TIP/30 system.

"Section 3 TIP/30 Utilities"

This section contains descriptions of the TIP/30 utilities, arranged in alphabetic order.

"Glossary"

This section contains definitions of TIP/30 and data processing terms and concepts.

"Index"

This section contains an index for this volume of the manual.

Results

After reading this document, your data processing staff will have a general understanding of TIP/30 concepts and will be able to determine which TIP/30 utilities are best applicable to your data processing needs.

Contents

Contents

Section 1
Overview of TIP/30

TIP/30 is an integrated system of transaction processing and program development software which runs under the control of the Unisys OS/3 operating system.

TIP/30 provides an environment that offers the following advantages:

- TIP/30 facilitates the development of application systems
- TIP/30 has a large number of productivity tools
- TIP/30 makes most efficient use of hardware resources
- TIP/30 executes existing IMS action programs without modification (no need to compile or link IMS programs).

Because it is a complete and comprehensive software system, TIP/30 represents the most powerful transaction processing and program development software available to the OS/3 user.

The heart of the TIP/30 software system is a multi-thread Program Control System, an integrated Message Control System, and a comprehensive File Control System. Included in this nucleus is an extensive system access security control facility as well as facilities for maintaining user data base integrity.

In addition, TIP/30 includes an extensive library of interactive utility programs to aid in program design, testing, implementation and system monitoring.

1.1. Program Control System

The TIP/30 Program Control System provides multi-thread control of application programs.

Through a concept of program stacking, TIP/30 provides inline returns from all program CALLs. This feature allows each individual program to do more work, thereby reducing the number of programs required in an online system.

TIP/30 allows the application designer a great degree of flexibility in the design of applications.

The TIP/30 Program Control System allows application designers to concentrate on the application instead of ways to overcome constraints imposed by the software methodology.

1.2. Message Control System

The TIP/30 Message Control System is an integrated facility that provides user programs and programmers complete freedom from terminal hardware characteristics.

MCS screen formats are developed interactively and stored in the TIP/30 screen format file.

TIP/30 provides a utility called MSGSHOW which is used to test developed screen formats with no programming required. Users can therefore participate in the design of screen formats. MSGSHOW makes it easier to develop online systems that feel comfortable to the user.

MCS allows programs to be written with no concern for the physical hardware characteristics of the terminal. The user program deals only with data. TIP/30 assumes the responsibility for knowing the hardware characteristics of the terminal. TIP/30 users can take advantage of new terminal hardware with no programming changes.

1.3. File Control System

The TIP/30 File Control System provides an efficient interface to all standard OS/3 Data Management files as well as integrated data base systems such as DMS (a Unisys product).

FCS provides record level and file level locking to preserve data integrity.

Automatic journaling of file updates provides online or off line recovery from system failures.

A system for creating, maintaining and scratching temporary scratchpad files allows flexible application design.

1.4. Security

System-wide security in the TIP/30 system is maintained through the TIP/30 catalogue file. All users, programs and files must be defined in the TIP/30 catalogue before they can be referenced online. TIP/30 guarantees security for a user, his programs and his files.

Users can be required to logon to TIP/30 with a specific user id and password. The password for a user cannot be displayed by anyone (not even the system administrator).

A horizontal layering of security is achieved by the use of a security level number in the range of 1 to 255. A user can only access those system facilities permitted by his catalogued security level.

A vertical partitioning of users, programs and files by application group can be achieved through the group specification in the TIP/30 catalogue.

A user logged on the TIP/30 system with a valid password only has access to those features of the system belonging to his application group for which his security level is high enough to permit him access.

1.5. Interactive Utilities

As a TIP/30 user you have access to an extensive library of interactive programs to assist in the design, implementation, testing and maintenance of online application systems.

1.6. Program Preparation

For program preparation TIP/30 provides a powerful text editor, an online librarian and a spool file inquiry utility.

The TIP/30 text editor is used to create and modify text elements. These elements may be program source, job control statements, or documentation.

The editor's work space is fully recoverable so that no work is lost due to a system failure. This feature results in higher morale and greater productivity on the part of the programming staff.

1.7. Screen Format Preparation

TIP/30 Message Control System screen formats are created and maintained online by the screen format definition utility (TFD).

Screen formats can be tested online using the MSGSHOW utility. The screen format maintenance utility, MSGAR, can be used to print screen formats, save screen formats in an OS/3 library, and to restore screen formats that were saved in an OS/3 library.

Creating and maintaining screen formats in a TIP/30 system is a very simple task. User departments can work with the development staff to design "friendly" screen formats to help ensure online system success.

1.8. Program Testing and Debugging

TIP/30 provides utilities to help the programmer get his programs running quickly.

Program dumps may be displayed online by the Post Mortem Dump Analysis program (PMDA). This eliminates waiting for the central printer and may also reduce the need to print lengthy dumps.

User programs can be traced online by the Interactive Debug Aid (IDA). A programmer can set break points in a program or trace a program one machine instruction at a time. Errors can be corrected at execution time and the execution of the program can be resumed. A programmer can find more errors during each program test using IDA and reduce the number of compilations required to implement a program.

1.9. Utilities

The TIP/30 system includes batch utility programs to initialize and backup TIP/30 system files as well as a program to analyze and summarize the system accounting information that may be maintained in the TIP/30 journal file.

Section 2
Fundamental Concepts

2.1. User Identification and Passwords

A user of the TIP/30 system is assigned a "userid" by the installation administrator. This userid is intended to be a meaningful pseudo name for the individual. It often takes the form of the individual's last name, his initials, or any character string of up to eight characters (starting with a letter of the alphabet) that serves to identify the individual within the TIP/30 user community. For example, user "John Q. Doe" might have a userid of "DOE", "JOHN" or "JQDOE".

To be able to enforce system security, the TIP/30 system must be able to verify that an individual is who he claims. To that end, there is a "password" associated with every userid in the system. Each user is assigned an initial password with his userid.

The user should be aware that the password is an agreement between the user and the TIP/30 system on a means of positively identifying the user.

The TIP/30 system does not provide any way to display a user's password! If a user "forgets" his password there is absolutely no way anyone — even the system administrator — can determine what the password is. The only possible course of action is to have the system administrator assign a new password for that user.

While it is intended that the individual's userid be known to other users, the password is the first and most fundamental level of security. The password should not be known to anyone but the individual.

The TIP/30 system only requires a password to be given at logon time; the password is not required to run programs or to access files. Once a user has logged on the system his capabilities are well defined by his positive identification.

Associated with each user of the system is their security level. Each user is assigned a security level by the installation administrator. This security level is a numeric value from 1 to 255.

In general terms, the security level is a statement about the access the user may have to programs and files. A numerically low security level indicates that the user has a high degree of access. Since there are 255 security levels, users may be easily organized into logical access groups. A user may not access a program or file if their security level does not permit access.

TIP/30 users may be given membership in groups. These user groups are established by the installation administrator. A user is a member of two implicit groups: his own private group (with the same name as his userid) and the system-wide group (named "TIPY").

Each user may also be given membership in one or two optional or elective groups. The installation administrator specifies the elective group memberships at the time a userid is established. These elective group memberships may be changed at any time by the administrator.

Membership in a group grants the user the potential to access programs and files belonging to the group. Actual ability to execute a specific program or file depends on the security level of the user with respect to the security level of the program or file in question.

2.2. LOGON and LOGOFF Procedures

A user must LOGON the TIP/30 system in order to identify himself to the system and to establish his capabilities with respect to the TIP/30 security system. There are two methods to LOGON:

- immediate transmission of a valid userid and password
- transmission of anything not valid as a userid and password.

If the user transmits (at an idle terminal) a valid userid and password he is logged on immediately. For example, a user with a userid of "FRED" and a password of "QWERTYUI" might logon by transmitting either of the following:

```
FRED/QWERTYUI
```

```
LOGON FRED/QWERTYUI
```

Note the required character (slash) separating the password from the userid.

If the user transmits an invalid userid/password combination or presses a function key or msg-wait, the TIP/30 system responds by displaying a screen format similar to the one shown below:

```
TIP/30   LOGON    FRIDAY JULY 13 1984            Time: 10:01

               Site: -site--name-

                     Please LOGON

          Userid : ________
        Password :
  Account Number : ____

  Place cursor here  (_)  and press XMIT
```

The user is expected to fill in the appropriate values and press the XMIT key. TIP/30 validates the userid and password. The account number field may be required — this is a configuration option controlled by the system administrator for each site.

The user is presented with this screen format for up to 4 attempts. (The user must press [XMIT] within 60 seconds, or the LOGON program clears the screen and the LOGON procedure has to be started over).

When the user has successfully logged on the system, TIP/30 displays the standard system prompt:

```
TIP?▶
```

The text that is displayed as the standard system prompt is a TIP/30 generation (configuration) option that may be customized by the installation. The value shown above is the default prompt text.

To log off the TIP/30 system, the user may run the program "LOGOFF" as a response to the standard system prompt. To run the LOGOFF program the user enters:

```
TIP?▶LOGOFF
```

The logoff program terminates the session and outputs a display giving the date and time of logoff and various statistics about the session that was just completed. The statistics include average response time and the number of input and output messages to the terminal.

The installation administrator may (at his discretion) change the name of the "LOGON" and "LOGOFF" programs. Users are advised to review their installation's LOGON and LOGOFF requirements with the installation administrator when they receive their userid and initial password.

2.3. TIP/30 Command Line

The TIP/30 system displays the standard system prompt on the terminal after a successful logon and whenever control returns to TIP/30 from a transaction program. In order to run transaction programs, the user must be familiar with the structure of the command line. When the system issues the standard system prompt the terminal user has an opportunity to enter a command to the TIP/30 system. This command has the following structure:

```
TIP?▶trid[,options] [parameter1] ... [,parameter8]
```

The system prompt is a configurable string followed by a start of entry (SOE) character. The system prompt shown above is the default system prompt.

The transaction-id (trid) immediately follows the start-of-entry character (▶) and represents the name of the program the user wishes to run. The trid may be up to eight characters long.

Some transactions (programs) allow the user to enter options immediately following the transaction-id. The options are separated from the trid by a comma or a slash. Options are from one to eight characters which are defined by the particular transaction program.

Separated from the transaction-id (and possibly the option characters) by at least one space are parameters for the program. There may be up to eight parameters supplied to the program from the command line.

These command line parameters represent initial input data for the program. Each parameter is restricted to a maximum of eight characters. The parameters are positional; any omitted parameters are indicated by the presence of a comma separator without any data.

Example:

```
TIP?▶PAYROLL MAR,,1982
```

In the example above "PAYROLL" is the transaction-id; there are no command line options; parameter-1 is "MAR"; parameter-2 is omitted; parameter-3 is "1982".

It is important to note that the parameters on the command line are passed to the indicated program as initial data. The programmer who wrote the program is free to interpret the parameters in any manner he chooses. The TIP/30 system merely enforces this command line convention as a simple means of running a program.

It is quite reasonable for a program to require no information from the command line (for example, menu-oriented systems frequently require that the user enter only the name of the menu program).

Many of the utility programs supplied with TIP/30 make extensive use of the command line options and parameters. The documentation for these programs describes the command line parameters recognized by each utility and the command line options required.

2.4. TIP/30 System Security

TIP/30 provides an extensive security system that may be utilized by the installation administrator to control access to programs and files. The security system cannot be selectively disabled or circumvented. The security system is implemented by entries in the TIP/30 catalogue. The catalogue is a TIP/30 file that is managed by the online catalogue manager program (CAT). The installation administrator uses the catalogue manager program to enter and modify information in the catalogue.

The catalogue contains entries for all authorized users of the TIP/30 system, all programs that are available online, and all files that are accessible online.

Each authorized TIP/30 user has an entry in the catalogue that states his userid, current password, security level and the names of (up to sixteen) elective groups to which he belongs.

There is an entry in the catalogue for each program (transaction-id) which states the group to which the program belongs and the security level required to access the program.

There is an entry in the catalogue for each online file in the system. The entry indicates which group has access to the file and the security level required to access the file.

A user may only access programs and files that are defined in the group(s) in which the user is a member. Furthermore, even though the user is a member of a group his access to programs and files in that group is restricted further by the requisite security level.

The catalogue manager program is generally assigned a high security level so that only users with high security (the installation administrator) may change entries in the catalogue. This ensures there is no mechanism whereby the average user can alter security levels or group memberships.

When a user attempts to run a program or access a file, TIP/30 searches for a corresponding program or file entry in the TIP/30 catalogue. The search follows a fixed order, known as "the order of search". TIP/30 searches the user's private group, any elective groups, and the system universal group — TIPY. The first program or file entry that is found is considered to be the intended one. The user's security level is compared to the required security level to run the program or access the file.

If the security level does not imply access, TIP/30 displays a "SECURITY ERROR" message. It is very important to note that the catalogue search does NOT continue past the first entry found in the predefined order of search. If no appropriate entry is found in the catalogue, the user receives an error message stating that the program or file could not be found.

The above description of search order also applies whenever a program attempts to call another program or access files. The catalogue is searched every time an attempt is made to access a program or file. The TIP/30 catalogue file is organized in such a fashion that this order of search may be accomplished very quickly. There is no appreciable overhead associated with this security mechanism.

Section 3
TIP/30 Utilities

3.1. ACCESS — Access File

The ACCESS transaction is used to manually assign a Logical File Name (LFN) to a file and to make the file available to programs executed at the terminal. Most transaction programs access files by specifying the logical file name in the catalogue entry for the program or by calling the TIP/30 File Control System (TIPFCS) with the FCS-OPEN function.

Syntax:

```
ACCESS[/type]   aftname,lfn
```

Where:

type Command line option: "P" or "T" indicating whether the file to be accessed is a permanent or temporary file (only applicable if the file is a TIP/30 Dynamic File).

aftname The Logical File Name to be assigned to the file. This is the name used in the active file table and is the name used by programs to identify the file to the TIP/30 File Control System (FCS).

lfn The catalogue name of the file. If the file to be accessed is an FCS dynamic file, the catalog-name consists of three sections: USER-ID, CATL-ID, FILE-ID.

If the USER-ID is not specified, then the USER-ID used to LOGON to TIP/30 is used. If left blank then CATL-ID is set to the value of FILE-ID.

In the following example assume that user 'ARC' is logged on.

Example:

```
ACCESS  UPDATE,MASTER
```

This example assigns the file with catalogue name 'MASTER' to the issuing user under the name 'UPDATE'. The AFT (user's Active File Table) would look as follows:

Logical	User-id	Catl-id	File-id	Type	Class	Hold	Element
UPDATE	ARC		MASTER	DIRAM	S	UP	

3.2. ALLOC — Allocate OS/3 File

The ALLOC transaction allocates an OS/3 file. The user specifies the name of the file, the initial and secondary space allocation, the file type, and the volume serial number of the disk where the file is to be located.

Syntax:

```
ALLOC  FIle=  TYpe=  SIze=  [VSn= ]  [INc= ]  [CONtig= ]
```

Where:

FIle=	The LBL name of the file to be allocated. This keyword is required. LBL names containing an embedded space or period must be enclosed in quotes.
TYpe=	The type of file to be allocated. This keyword is required. Specify one of: ST, MI, IS, DAM, SQ, or NI.
SIze=	The number of CYLINDERS to allocate for the file. This keyword is required.
VSn=	The volume serial number of the disk volume where the file is to be allocated. This parameter may be omitted if the file has already been catalogued in the OS/3 system catalogue (YCAT).
INc=	The secondary increment (in cylinders) for this file. Default is a secondary increment of 1 cylinder.
CONtig=	Specify CONtig=YES or CONtig=NO (the default) indicating whether or not the allocated disk space must be contiguous.

Example:

```
ALLOC FILE='TEST.FILE' VSN=REL130 TYPE=MI SIZE=8
```

This example allocates a file with an LBL name of "TEST.FILE" on the volume REL130. This file is allocated as MIRAM with an initial space allocation of 8 cylinders. The file would not necessarily be allocated as contiguous cylinders (since CONTIG=YES was not specified).

3.3. APB — All Points Bulletin

The APB program sends a broadcast message to all currently active terminals (except the sender).

The message is sent as an unsolicited message. When the message is received, it will be prefaced by the userid and terminal name of the sender.

Syntax:

```
APB[/ALL]      text
APB[/grp]      text
```

Where:

ALL — Command line option indicating that the APB message is to be sent to all terminals connected to TIP/30 (whether or not the terminal is logged on TIP/30).

If this option is not specified, only terminals that are currently logged on TIP/30 receive the message.

This command line option is coded as the three characters "ALL".

grp — Command line option indicating that the APB message is to be sent to terminals where the logged on users are members of the specified user group.

text — The text of the message (64 characters maximum). The text need not be enclosed in quotes.

Example:

```
APB  SYSTEM WILL SHUTDOWN AT 21:15 FOR 30 MIN.

APB/EDP Meeting at 15:30 in boardroom
```

3.4. ASG — Assign a File

The ASG program is used to assign a logical file name (LFN) to a file and to make the file available to programs executed by the process. Most transaction programs access files by specifying the logical file name in the catalogue entry for the program or by calling the TIP/30 File Control System (TIPFCS) with an FCS-OPEN function.

If the ASG program is used to assign a TIP/30 Dynamic File that does not currently exist, the dynamic file will be created.

Syntax:

```
ASG[/type]   lfn,filename
```

Where:

type "P" or "T" indicating whether the file to be accessed is a permanent or temporary file (only applicable if the file is a TIP/30 Dynamic File).

lfn The Logical File Name assigned to the file. This is the name used in the active file table and is the name used by programs to identify the file to the TIP/30 File Control System (FCS).

filename The catalogue name of a file. If the file to be accessed is an FCS dynamic file, the catalogue name consists of three sections:
USER-ID/CATL-ID/FILE-ID

If the USER-ID is not specified, the userid used to LOGON to TIP/30 is used. If the CATL-ID is omitted then CATL-ID = FILE-ID.

In the following example assume that user "ARC" is logged on.

Example:

```
ASG,P   TEST,,TESTFILE/ONE
```

This creates a TIP/30 Dynamic File called TESTFILE/ONE and assigns it with the logical file name "TEST". The active file table for the terminal would then look like this:

```
Logical  User-id  Catl-id  File-id  Type  Class Hold Element
=======  =======  =======  =======  ====  ===== ==== =======
TEST     ARC      TESTFILE ONE      DYNAM  P
```

3.5. B* — Begin All Spool for a Job

The B* utility submits an OS/3 console command to begin ALL spool queues for a specified job. The generated command is logged on the OS/3 console and in the job log for the TIP/30 job.

The command submitted is: **BE SPL,ALL,JOB=xxxxxxx** unless special processing options are requested.

Syntax:

```
B*[,opt]  [jobname]
```

Where:

opt Command line option that may be used to specify either a specific printer device type or a specific printer address.

If the option field begins with the character "7" (for example: B*,770) the option field is interpreted as a printer device code.

If the option field begins with a character other than a "7" (for example: B*,3E0) the option field is interpreted as a printer device address.

The actual OS/3 command that is generated varies according to the value in the command line option field. See the examples shown below.

jobname An optional job name that is to be used in the generated OS/3 command.

If this parameter is omitted, the B* program uses the job name of the currently executing TIP/30.

Example:

Command Line	Resulting Command
B* COB74	BE SPL,ALL,JO=COB74
B*,770 FRED	BE SPL,ALL,JO=FRED,DEV=770
B*,3E0 FRED	BE SPL,PR,JO=FRED,OUT=3E0

Additional Considerations:

Refer to the description of the OS/3 operator console command "BE" in the operation guide for your system.

3.6. BE — OS/3 BEGIN Command

The BE transaction implements the OS/3 "BEGIN" console operator command. The BE transaction is actually a clone of the more powerful SYM transaction. The SYM transaction program can be invoked using an alias name that is interpreted as a specific command. When the SYM program is invoked with a transaction name of "BE", the SYM program assumes that the OS/3 command is "BE".

The OS/3 BEGIN command syntax is documented in the operation guide for your system.

The BE transaction requires the same syntax as the console command. The command is passed exactly as entered to the OS/3 command processor for execution as if it was entered at the system console.

Note: *There is no provision for returning any completion status.*

Example:

```
BE COBCOMP3        { begin a queued job }
```

The command shown in the example issues a BEGIN operator command for a job named "COBCOMP3".

3.7. BR — OS/3 BR Command

The BR transaction implements the OS/3 "BR" console operator command. The BR transaction is actually a clone of the more powerful SYM transaction. The SYM transaction program can be invoked using an alias name that is interpreted as a specific command. When the SYM program is invoked with a transaction name of "BR", the SYM program assumes that the OS/3 command is "BR".

The OS/3 BR command syntax is documented in the operation guide for your system.

The BR transaction requires the same syntax as the console command. The command is passed exactly as entered to the OS/3 command processor for execution as if it had been entered at the system console.

Note: *There is no provision for returning any completion status.*

Example:

```
BR PR,ACT,JO=TIP30
```

The command shown in the example breakpoints the active print file for the job named TIP30.

3.8. BRKPT — Breakpoint Print File

The BRKPT transaction causes a breakpoint to be taken for a print file that is defined to TIP/30. The print file is closed to create a physical breakpoint.

Syntax:

```
BRKPT  lfn
```

Where:

lfn — The logical file name of a printer file that is defined to TIP/30. The logical file name is the name that is defined in the TIP/30 Catalogue for the file (not necessarily the LFD name). The file must be defined in the TIP/30 generation parameters as type "PRINT".

Example:

```
BRKPT PRNTR
```

Note: *The resulting status is displayed: "Print file breakpointed" or "Print file not breakpointed". The latter message text may be the result of misspelling the logical file name (LFN) or may result if the LFN is not defined or is not a print file.*

3.9. BX — OS/3 PR BX Command

The BX transaction implements a variation of the OS/3 "PR" console operator command (start burst mode output writer). The BX transaction is actually a clone of the more powerful SYM transaction. The SYM transaction program can be invoked using an alias name that is interpreted as a specific command. When the SYM program is invoked with a transaction name of "BX", the SYM program assumes that the OS/3 command is "PR BX".

The OS/3 PR command syntax is documented in the operation guide for your system.

The BX transaction is a shorthand notation for the more cumbersome syntax of the "PR BX" console command. The BX transaction accepts a job name and submits an appropriately formed "PR BX" command to begin a burst mode output writer for the supplied job name.

Note: *There is no provision for returning any completion status.*

Syntax:

```
BX  [jobname]
```

Where:

jobname An optional command line parameter that is used to specify the target OS/3 job name.

If omitted, the BX transaction will use the name of the TIP/30 job that is executing.

Example:

```
BX  PAY020
```

This example command results in the OS/3 command: PR BX,JO=PAY020

3.10. CA — OS/3 CANCEL Command

The CA transaction implements the OS/3 "CANCEL" console operator command. The CA transaction is actually a clone of the more powerful SYM transaction. The SYM transaction program can be invoked using an alias name that is interpreted as a specific command. When the SYM program is invoked with a transaction name of "CA", the SYM program assumes that the OS/3 command is "CA".

The OS/3 CANCEL command syntax is documented in the operation guide for your system.

The CA transaction requires the same syntax as the console command. The command is passed exactly as entered to the OS/3 command processor for execution as if it had been entered at the system console.

Note: *The CA transaction does not allow an ICAM symbiont to be cancelled. The CA command cannot cancel the TIP/30 job that is issuing the CA command.*

Example:

```
CA COBCOMP3,N
```

This example command cancels job "COBCOMP3" without a dump.

3.11. CAT — TIP/30 Catalogue Manager

The CAT program is a system utility which displays, updates, adds, or deletes records in the TIP/30 catalogue. The CAT program operates interactively and accepts commands which have a very flexible syntax.

The user enters a command code, one or more positional parameters (used to identify the required item), and keywords to supply information about that item.

CAT also accepts command line parameters for the list functions.

Syntax:

```
Command  P1,P2,P3,P4  key1= key2= key3= ... keyn=.
```

Where:

Command The CAT function to be performed. Valid functions are:

Table 3-1. CAT Commands

Command	Description
List	List catalogue record(s) on the terminal.
Write	Write catalogue record(s) to a library source element.
DELete	Delete catalogue record(s).
Prog	Create/Update a program record.
User	Create/Update a user record.
File	Create/Update a file record.
Group	Create/Update a groupset record.
End	Terminate the CAT program.
Quit	Terminate the CAT program and logoff TIP/30.

P1,P2,P3,P4

Positional parameters supplied with the command.

For the Prog, User, File, and Group commands, only the first two positional parameters are used — positional parameters three and four need not be specified.

For the List, Write, and DELete commands, positional parameters one through four are used to identify the item(s) to be processed.

key1= ...keyn=

Keyword parameters that are used to supply additional information.

3.11.1. TIP/30 Catalogue File

The TIP/30 catalogue file (TIP$CAT) contains the information needed by TIP/30 to execute online programs. The catalogue is organized as a hashed file (the key of a record is transformed into a relative block number which indicates the place to store the record). Specific records can therefore be retrieved without searching an index.

All of the information in the TIP/30 catalogue is available to users immediately when updated; maintenance is performed by an online program. The exception to this rule is user entries. The catalogue entry of a currently logged on user is not updated when the user record is changed until the user logs off and on again or uses the NEWUSER utility.

Changes to the catalogue information is effective immediately and such changes are directly updated in the file.

The TIP/30 catalogue contains the following types of records:

User These records identify valid users of the TIP/30 system.

Program These records describe valid online (transaction) programs.

The program record identifies all the run time requirements of the program such as: load module name, MCS area size, Work are size, etc).

File These records describe valid online files.

The file record in the catalogue links the logical file name (the name used in a program) to the LFD name (the name used in the operating system).

Groupset These records define SETS of group names.

A user may be given membership in several elective groups - if the user belongs to more than two elective groups, they must be specified in a groupset record.

Each record in the catalogue has a 25 character logical key. This key is composed of four fields which together uniquely identify each record in the catalogue.

Duplicate keys in the catalogue are not allowed.

The four fields that form the catalogue key are as follows:

Group This is the name of the group to which the item belongs.

In the case of a userid record, this field contains the userid.

In the case of a Groupset record, this field is the name of the groupset.

Any items (programs or files) catalogued in a group with the same name as a userid record, are considered to be in that user's private group.

Id If the catalogue record is a file type record this field contains the logical file name (LFN) of the file.

If the catalogue record is a program type record this field contains the TRID (transaction id or program name).

If the record is a userid type record or groupset type record, this field is essentially not used.

Elt This field is only used if the catalogue record describes a TIP/30 dynamic file.

Dynamic files have a two level name (id/elt), and this field is used to contain the element name of the dynamic file.

Type This is the type code of the catalogue record. There are four type codes used as follows:

U	User record
P	Program (TRID) record
F	File (LFN) record

File records in the catalogue may also be referenced by the class code which identifies the type of file as follows:

S	System (data management) file
D	Dynamic file
E	Edit buffer
G	Groupset record

Since the catalogue is a hashed file (has no index) it cannot be processed in any specific order other than sequentially by physical block.

To produce an ordered listing of the file (either online or in batch), the file must be entirely scanned at the block level to extract the desired records, the records selected must be sorted to produce the desired listing.

An understanding of this fact facilitates obtaining listings of records in the catalogue in an efficient manner.

The TIP/30 catalogue file (LFD=TIP$CAT) is implemented as a single partition SAT file.

The catalogue file must not be processed by any programs other than those supplied with TIP/30 or the standard operating system file dump/restore program (DMPRST).

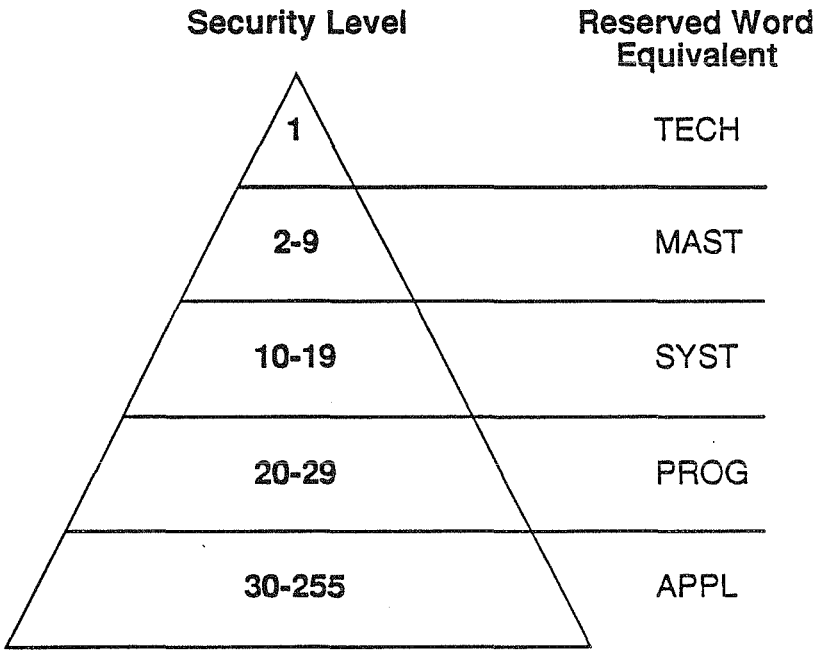
3.11.2. Security Levels

The SECurity= keyword is common to the USER, PROGRAM, FILE and GROUPSET catalogue records. This keyword specifies information that is needed by the TIP/30 security system.

A user has a specific security level that he has been granted by the system administrator. A user's security level implies that he may access a program, file or groupset only if the item has a security level numerically not greater than the user's security.

A PROGRAM, FILE and GROUPSET record in the catalogue specifies the required security level that a potential user must have to access or use the item.

Security levels range from 1 (the highest security clearance) to 255 (the lowest security clearance):



Users at a particular security level are able to access PROGRAMS, FILES and/or GROUPSETS that have a security level requirement **at or lower** than the user's security level (as discussed shortly, this statement assumes proper group membership beforehand!).

The SECurity= keyword may be specified in two different ways. Either an absolute numeric value may be given, or a reserved word may be used:

Table 3-2. Catalogue SECURITY Specification

Reserved Word	Numeric Value
SECurity=n	Determines the security level of the catalogue record. Enter a numeric value between 1 and 255 inclusive.
SECurity=TECH	Equivalent to SEC=1.
SECurity=MAST	Equivalent to SEC=9.
SECurity=SYST	Equivalent to SEC=19.
SECurity=PROG	Equivalent to SEC=29.
SECurity=APPL	Equivalent to SEC=32.

The following table lists the security ranges provided and the CAT commands allowed by the online CAT program for users in each security range.

Table 3-3. CAT Capabilities by Security Level

Security	Functions
1 (TECH=1)	System Administrator: — may create any other level users — may list, create, update, delete any entry in the catalogue.
2-9 (MAST=9)	Master User: — may list, create, update or delete any record in the catalogue except user records of users with higher security.
10-19 (SYST=19)	System User: — may list, create, update or delete any catalogue record in a group to which he has access (is a member) — may create user records, but may not grant access to a group that the system user does not have access to — may not create or update group sets.
20-29 (PROG=29)	Programmer User: — may list any catalogue record in a group to which he has access — may create, update or delete program records in his private group — may not create or update file, user, or groupset records.
30-255 (APPL=32)	Application user: — not allowed to access the catalogue in any manner.

Note: *The rules described in the preceding table are enforced by the CAT program. The CAT program does not allow these rules to be circumvented (even by altering the security level of the CAT program).*

When cataloguing FILES, PROGRAMS and GROUPSETS, assign a security level numerically equal to or less than the security level of the user or users who need to access the item.

When a user attempts to access a file or to run a program the following security check takes place for these items:

```
IF user security level > item's security
   THEN deny access
ELSE
  IF item is time-locked
     THEN  deny access
  ELSE  allow access
  END-IF
END-IF
```

Note the comparison is a strictly numeric comparison of the user's stated security and the item's stated security.

Example of Security Clearance Comparison:

User Security	Item Security	Access?
29	50	Yes
29	29	Yes
29	19	No
1	any	Yes

3.11.3. Definition of User Groups

The concept of grouping in the TIP/30 catalogue must be understood to properly utilize the catalogue and its features. Every program and file is defined within a group (a program or file may appear in several groups).

Every user of the system has a list of groups to which the user has access.

When a user requests access to a program or file, each group to which the user has access is consulted to determine whether the requested item is defined in that group.

The order the groups are consulted is known as the catalogue **order of search** and is defined as follows:

Private This is the user's private group.

Any items (programs or files) catalogued in a group with the same name as the userid name are considered to be in the private domain of that user.

Group 1 This is an optional (elective) group and is consulted if the userid record was created using the GRouP=(a,b) keyword parameter.

The first sub-parameter of the GRouP= keyword is used to identify elective group one.

Group 2 This optional (elective) group is similar to Group 1 above. It is the third group to be consulted when searching for an item.

The second sub-parameter of the GRouP= keyword of the User command identifies elective group two.

groupset Optional, additional elective groups that extend the elective groups beyond Group1 and Group2.

TIPY This group is the last group consulted in the order of search. The name of the group is the reserved name "TIPY".

Note: *Group membership and the order of search control only which item in the catalogue is selected for a given user.*

It is the security level of the item selected that ultimately determines whether the item may be used by that user.

It is through the specification of groups and security levels that the TIP/30 security mechanism is able to control user access to programs and files.

3.11.4. Group Security Levels

When TIP/30 searches the TIP/30 catalogue to resolve PROGRAM and FILE references, the user's stated security is ultimately used to determine whether an item is available to that particular user.

It is possible to specify a security level that is associated with an elective group. This security level effectively overrides the user's security when the security check for that group is performed.

Example:

```
USER  FRED  SECUR=PROG  GRPS=(ACCTNG,55,PAYROLL,65) ....
```

The effect of the example user specification is the following:

- FRED is a programmer level user
- FRED belongs to the following groups: FRED, ACCTNG, PAYROLL, TIPY
- when performing the security check for PROGRAM or FILE entries that are found in the group **FRED** or **TIPY**, FRED's security is PROG (29)
- if a PROGRAM or FILE entry from the GROUP **ACCTNG** is involved, the specification **55** overrides the value in the SECUR= keyword FRED has level 55 security when searching the group ACCTNG;
- if a PROGRAM or FILE entry from the GROUP **PAYROLL** is involved, the specification **65** overrides the value in the SECUR= keyword — FRED has level 65 security when searching the group PAYROLL.

Very important point:

The use of the security level associated with elective groups is limited to determining the user's access to items in the group. In the example above, FRED is a programmer level user. He happens to have diminished access (less than a programmer) in his elective groups.

If a security level is omitted for an elective group, the user's security is assumed.

Syntactical note: the security level that may be specified with the group name in the GROUPS= keyword must be specified as a numeric value — the short form names (TECH, MAST, SYST, PROG, APPL) are not allowed because they could be erroneously interpreted as a group name.

Example:

```
GRPS=(EDP,PAYROLL,55)
GRPS=(EDP,110,PAYROLL)
GRPS=(EDP,PAYROLL)
```

3.11.5. Groupsets

A groupset can be considered to be a definition of a pool of "additional" elective groups. A groupset may be chained to another groupset record to create very long chains of potential group names.

A groupset may be created that establishes a number of elective groups (and corresponding security levels). Once this groupset is defined and given a name, users may be assigned to have access to the named groupset.

The GROUPS transaction (see "3.40. GROUPS — Modify Elective User Groups" on page 3-151) may be used to dynamically reorder the elective groups to temporarily override the usual order of search.

Altering the user's elective groups via the GROUPS transaction is a temporary change to the user's elective group membership; all alterations are subject to a check that the requested group names are part of the groupset to which the user belongs.

Example:

```
USER  FRED  SECUR=29 GROUPSET=SALES GRPS=(EDP,GOLFER) ...
GROUPSET SALES SECUR=PROG GRPS=(MKTNG,55) GRPS=(BUDGET,65)
```

In this example, user FRED is defined to be a member of the groupset SALES.

At logon, FRED's elective groups are EDP and GOLFER (and appear in FRED's order of search in that order).

The security associated with groups EDP and GOLFER is 29 (since it wasn't explicitly specified); the security associated with groups MKTNG is 55 and BUDGET is 65.

Therefore, if FRED uses the GROUPS utility and enters:

```
TIP?▶GROUPS MKTNG,BUDGET
```

the effect is to temporarily alter FRED's order of search to be as follows:

```
MKTNG   BUDGET
```

The groupset record allows the system administrator to define up to 16 groups and corresponding security levels that represent additional elective groups for users who are assigned to that groupset.

3.11.6. USER — Cataloguing a Userid

The catalogue manager **USER** command creates or updates a userid record in the TIP/30 catalogue. If the specified user name already exists in the catalogue, the command is considered an "update" operation rather than an "add" operation.

Syntax:

```
User  userid  kwd= kwd= ...
```

Where:

userid Required positional parameter!

This parameter specifies the userid of the user record being created or updated.

A maximum of eight characters may be specified.

After the userid, one or more keywords follow:

Table 3-4. USER Definition Keywords

Keyword	Description
ACcounTs=	List of valid accounts for this user.
CoMmenTs=	User's name or other identification.
DeBug=	Set user in "test" mode.
DFLTACCT=	Default account number.
EXPIRY=	Date when this userid record "expires".
GRouPs=	User's elective group memberships.
GRouPSeT=	Define user's group set.
LANGuage=	Specify user's language code.
LOGonSeT=	Define user's initial elective groups.
MaXUSers=	Maximum concurrent logons with this userid.
MCSearch=	Type of screen format searching.
MENU=	Screen format to be used as menu.
NCS=	OFIS Link National Character Set.
PassWorD=	TIP/30 logon password for this user.
PRoGram=	Program automatically called at LOGON.
SeaRCH=	Restrict standard order of search.
SECurity=	User security level.

continued ...

Keyword	Description
SP=	Set destructive space bar.
(period)	Signal end of keywords for user command.

ACcounTs=(,,,)

A list of valid accounts that may be supplied by this user when logging on TIP/30. Up to 16 account "numbers" can be specified.

Each account "number" may be a character string of up to four characters (not necessarily numeric).

If this keyword is specified the user **must** supply one of the valid account numbers when logging on TIP/30.

If only one account number is specified here it is used as a default account for the user (if the user chooses to omit the account number at logon).

The account number appears in logoff information that is written to the TIP/30 Journal or Log file (if configured).

CoMmenTs=

Up to 28 characters of descriptive information which is associated with this user.

The data **must** be placed within single quotes if it contains an imbedded space or period.

A popular choice for this data is additional information like the user's full name and telephone extension.

The utility program USERS (see "3.94. USERS — Display User Directory" on page 3-355) displays this comment information.

The first 19 characters of this comment string appear in any print header pages that are generated for the user by TIP/30 utilities.

DeBug= YES automatically sets the user into "test mode" at logon. In test mode, all updates to files are acknowledged, but not actually performed.

Default: NO (not in test mode).

DFLTACCT=

A four character account number that is taken as the default account number when this user logs on the TIP/30 system. If the account number field is left empty during the logon procedure, this account number is assumed.

EXPIRY= The date when this userid expires (when the userid is no longer accepted by the TIP/30 LOGON program).

Format is EXPIRY=yymmdd

This keyword may be useful when creating user ids for testing or for demonstrations.

GRouPs=(g1[,s1],g2[,s2])

This keyword identifies the user's initial (logon) elective groups in the situation where only one or two groups are desired (if more than two elective groups are desired the keyword LOGonSeT= must be used instead).

g1	The name of elective group 1
s1	Optional security level the user assumes when accessing a catalogue entry in g1 (default for s1 is the user's security as specified by the SECUR= keyword).
g2	The name of elective group 2
s2	Optional security level the user assumes when accessing a catalogue entry in g2 (default for s2 is the user's security as specified by the SECUR= keyword).

These optional (elective) groups are in addition to the user's private group and the system group (TIPY) used by TIP/30 to search the catalogue to resolve a reference to a program or file.

This keyword is optional; one or two elective groups (with or without security level) may be given.

This keyword and the LOGonSeT= keyword (described later) are mutually exclusive.

GRouPSeT=

The name of a GROUPSET record for this user.

A groupset record defines a list of (up to 16) of additional elective groups for this user.

GROUPSET records may be chained to other GROUPSET records to create a large "pool" of available group names. For more information, see "3.11.10. GROUPSET — Defining a Groupset" on page 3-39.

A user may dynamically alter his elective groups by using the **GROUPS** transaction.

LANGuage=

A one character (alphabetic) code that designates the language code for this user. This language code appears in the field PIB-LANGUAGE and may be interrogated by application programs.

LOGonSeT=

The name of a GROUPSET record that defines the initial (logon) elective groups (and any associated security levels) for this user.

This keyword and the keyword GRouPs= are mutually exclusive.

The groups defined via this keyword automatically are included in the set of available elective groups for this user.

Only the groups names defined in the first groupset record are considered the user's inital logon groups — that is, for this purpose, the NextGRPs= specification in the GROUPSET record is ignored.

MaXUSers=

Specifies the maximum number of concurrent uses of this userid.

Specifying MAXUSERS=1 implies that this userid may only be logged on one terminal at a time.

If this keyword is not specified, there is no limit to the number of concurrent logons with this userid.

MCSearch=

Specifies the type of screen format group searching for this user.

Specifying MCSEARCH=YES implies that the order of search (as defined in the SEARCH= keyword described following) applies.

Specifying MCSEARCH=NO implies that TIP/30 is to search **only** the TIPY group for screen formats.

Default is MCSEARCH=NO.

MENU= This is the name of a TIP/30 screen format to replace the standard system prompt for this userid.

If this keyword is not specified the standard system prompt is used.

When this keyword is specified, the TIP/30 command processor outputs the specified screen format instead of rolling the screen up one line and outputing the usual system prompt. Use of the screen format may (depending on how the screen format is defined) overlay all or part of whatever data was on the screen when the last transaction terminated.

NCS= Specifies (for an OFIS Link/80 userid) the desired National Character Set (see appropriate OFIS Link/80 documentation).

PassWorD=

Specifies the logon password for this user.

The password may be up to 8 characters in length and may contain any character that can be entered at the terminal keyboard.

A password that is the same as the userid or one which is symmetric (eg: "ABCDABCD") or one which begins with a space is considered an illegal password and is not allowed.

The password must be correctly supplied by the user when logging on TIP/30.

If this keyword is omitted, the user need not supply a password to logon TIP/30.

It is recommended that the password not contain imbedded spaces, or a slash or comma character.

Passwords which are entirely numeric are not recommended (numeric passwords are not compatible with the direct method of logging on TIP/30, since numeric values are right justified and zero filled — and would not match!).

PRoGram=

A transaction code which is to be automatically invoked (on the user's behalf) after a correct logon to TIP/30.

When this feature is used the user is automatically logged off when the specified program terminates.

This facility allows the user to be limited to a specific program (often a menu program) and therefore not allowing the user an opportunity to access other areas of the system.

If this keyword is omitted, TIP/30 does not automatically invoke any program when this user logs on TIP/30.

SeaRCH=

Controls the type of catalogue searching to be performed for this user.

If SEARCH=NO is specified, only the system group (TIPY) is searched.

If SEARCH=GROUP is specified, the user's private group (the userid itself) is NOT searched. That is, the order of search begins with the elective groups and continue through TIPY.

If SEARCH=YES is specified (or this keyword is omitted) a complete order of search (see previous discussion) is performed.

If this user never needs to access programs or files in a group other than TIPY, specifying SEARCH=NO eliminates the overhead involved in checking the private and elective groups.

SECurity=

The security level for this user.

This value controls the user's access to programs and files.

The security code may be specified as a number in the range 1-255 inclusive, or as one of the following reserved words: TECH(1), MAST(9), SYST(19), PROG(29), APPL(32).

The security level may not be specified as a (numerically) lower value than the security level of the user issuing the command.

Default: SECURITY=PROG.

SP=DS Specifies that this userid wishes to have the terminal control page altered at logon time to define the space bar as destructive.

This facility is only supported on terminals which have the SP/DS capability in the control page (U20, U30 U40 etc.).

This option may be specified on a terminal basis via a TIP/30 generation CLUSTER statement.

. (period) The last keyword specified as part of the USER command should be followed by a period (to indicate that the command is finished).

Example of dialogue defining a USER:

```
TIP?►CAT
CAT(1)?►USER FRED GROUP=EDP PWD=QWERTYUI SECUR=PROG
Continue USER statement► CMT='FREDERICK THE LESSER   x285'.
 USER catalogue record added  : FRED
CAT(1)?►
```

This example illustrates the addition (presumably) of a user who is assigned a userid of "FRED", a security of programmer level (29), and the indicated logon password.

The command was continued on an additional line (more on that in a later section) and the entire command was terminated by a period.

3.11.7. PROG — Cataloguing a Program

The catalogue manager **PROG** command creates or updates a program entry in the TIP/30 catalogue. All online programs (transactions) must be catalogued.

Whenever an area size is required (CDASIZE=, WORKSIZE=, etc), the value supplied must be a decimal number and represents the specified number of bytes.

To remove a sized area (when updating a program entry for example), it is necessary to specify the area with a length of zero (eg: CDASIZE=0).

Syntax:

```
Prog   group/trid  kwd= kwd= ...
```

Where:

group Required positional parameter to indicate to which group of users this particular definition of the program applies.

trid A required positional parameter specifying the transaction-id used to refer to this program entry. A transaction name may be from 1 to 8 characters in length (transaction names for programs running under IMS emulation are restricted by the value specified in the TIP/30 generation parameter: IMStranL=). IMS actions (that is, programs that are not invoked directly but are called internally by other programs) must be defined in the TIP/30 catalogue too; in this case, the trid is the same as the load module name because IMS requires the successor name to be the load module name.

Programs can be called only using their catalogued transaction-id (trid).

Transactions may be defined with special 4-character names of the form: F#nn (where nn is a two digit number 01 through 23 inclusive). These transaction names are specially interpreted at the TIP/30 command line to correspond to the associated function key.

For example, you may define a transaction named "F#03" to invoke your choice of load module. If a user presses F3 at the TIP/30 command line, the TIP/30 catalogue is searched to attempt to find a definition of a transaction program corresponding to F#03.

Such program catalogue definitions take precedence over any function keys defined using the DEFKEY utility transaction (see "3.21. DEFKEY — Define Function Keys" on page 3-73).

Table 3-5. PROG Definition Keywords

Keyword	Description
BaCK=	Program may run in Background.
BICS=	OFIS Link use BICS.
CDAsize=	Size of CDA required by program.
CMdLine=	Command line parameters required.
DeBug=	Type of debugging.
EDIT=	Removal of communications characters.
ENTer=	Allow execution from TIP/30 command line.
ERET=	Return error status to IMS program.
ESCape=	Allow escape from program.
FCCedit=	Edit FCC characters from input messages.
FiLes=	Files to be automatically accessed.
FRom=	Use keywords from another PROG entry.
IMS=	Program runs under IMS emulation.
INsize=	Size of IMA required (IMS emulation).
LoaDM=	Load module name.
MAXsize=	Maximum LOADM size in IMS internal succession or TIPJUMP.
MCSsize=	Size of MCS area required by program.
OUTsize=	Size of OMA required (IMS emulation).
PRIority=	Execution priority of program.
S34=	Simulate System/34 function keys for RPG II programs.
SECurity=	Security level required to run program.
SHRDsize=	Shared code size (alternate for VOL=).
SuBPRoG=	Definition is for a SUBPROG.
TiMeLocK=	Time range during which program may NOT be started.
TPmode=	Short-lived program.
TRANslat=	Translate input message to upper case.
TYPE=	Type of program (IMS or TIP).
USeage=	Program characteristics.
VOLatile=	Size of VOLATILE data area required.

continued ...

Keyword	Description
WoRKsize=	Size of WORK area required by program.
XMIT=	Control page XMIT requirement.
(period)	Signal end of keywords for PROG command.

BaCK=

BACK=YES implies that this program may run as a background process.

BACK=NO implies that this program may not run as a background process (attempts to do so from the command line fail with the error report: "Invalid transaction").

BICS=

BICS=YES indicates that this program requires BICS support.

This keyword is for OFIS Link programs only.

See TIP/30 OFIS Link documentation.

CDAsize=n

The CDA size (in bytes) required by this program.

TIP/30 does not place a restriction on the absolute size of the CDA (other than available memory).

The CDA is always allocated and manipulated by TIP/30 in multiples of 256 bytes. If the value specified in the catalogue for the CDA size is not a multiple of 256, the TIP/30 system interprets the value as the next higher multiple of 256.

CMdLine=

Specify whether or not the program is to have the parameterized contents of the command line as the initial contents of the CDA.

Default is "NO" for IMS programs; "YES" for TIP/30 programs.

Programs which are invoked via the TIP/30 command line and are catalogued with CML=NO begin execution with an input message outstanding (the "command line" input was not automatically read by TIP/30 — it is the program's responsibility to read this input message before soliciting terminal input).

DeBug=DEFAULT

The default setting of the DeBug= keyword.

This setting means that the program is to use hardware storage protection based on the current setting of the system-wide DEBUG option (a TIP/30 job control or TIPGEN parameter which defaults to "YES").

The SET utility program (see "3.78. SET — Alter Process Attributes" on page 3-244) may be used to turn the system debug setting on or off.

DeBug=YES

Specifies that this program is always to be run with hardware storage protection.

The program uses storage protection **regardless** of the setting of the system debug option.

This means that this program may abort with "PROTECTION EXCEPTION" if it attempts to modify memory outside its allocated space.

DeBug=NO

Specifies that this program is never to be run with hardware storage protection (regardless of the setting of the system-wide debug option).

Programs which modify the GDA require this specification.

DeBug=IDA

This program is to be loaded with the Interactive Debug Aid (IDA) in control.

Note that using IDA and running with hardware storage protection are mutually exclusive.

EDIT=NO

This is the default setting of the EDIT= keyword.

Input messages received by TIPTERM and IMS programs are NOT edited. DICE codes and FCC sequences are not removed from the input message.

EDIT=YES

Input messages received by TIPTERM or IMS emulation are edited to remove all communication control characters such as DICE and FCC sequences.

Multiple spaces are reduced to a single space.

EDIT=X'__'

Specifies a character to be used as the field separator in input messages received by TIPTERM or IMS emulation.

Multiple occurrences of this character are reduced to a single occurrence.

This character may be specified as a hex value (X'__') or as a simple character (eg: EDIT=%). The latter choice is only reasonable if the character is a graphic character.

This option also implies that DICE codes and FCC codes are to be removed from the input message (like EDIT=YES).

ENTer=

Whether this program may be called directly by entering the TRID on the TIP/30 command line.

Default: ENTER=YES.

Specifying NO implies that this program may only be invoked by transferring control to it from another program (ie: IMS succession, TIPSUB, TIPXCTL).

ENTER=NO is most often used in the definition of IMS actions (not to be confused with IMS transactions).

ERET=

This is a keyword for IMS programs that is the same as the IMS ERET= keyword which controls error return handling for IMS programs.

ESCape=NO

Specify whether or not the terminal user may escape to another program while this program is running (see description of the escape feature in the description of the TIP/30 Command Line).

Default: ESCAPE=YES.

FCCedit=

FCCEDIT=YES may be specified for IMS emulated programs to support the corresponding keyword from the IMS configuration parameters.

FiLes=(,,)

Up to 12 files to be automatically assigned to the program when the program is entered.

The filenames specified as sub-parameters for this keyword are the logical file names (not necessarily the LFD names).

Default: no files automatically assigned; the program must explicitly open files as needed.

This keyword may be specified more than once in a PROG command (if necessary) until the maximum of 12 file names is reached.

TIP/30 programs that need to open more than 12 files must open the additional files by issuing the appropriate call to the TIP/30 File Control System.

Programs that run under IMS emulation automatically acquire files as filenames are referenced by the program — files do not need to be identified in the catalogue program entry (although it is slightly more efficient internally if they are so defined).

FRom=grp[/trid]

Indicates that the keywords (except GRP=) for the program entry for "grp/trid" are to be copied for this program.

If the "trid" is omitted, it is assumed to be the same as the trid which is being specified.

IMS=YES

Specifies this is a catalogue entry for a program which is an IMS program (to be executed under IMS emulation by TIP/30).

INsize=n

The IMA size if this is an action program to be run under IMS Emulation.

LoaDM=

The name of the load module associated with this transaction-id.

Default is the same as the trid positional parameter.

Trailing zeroes need not be specified.

MAXsize=n

For programs running under IMS emulation and utilizing immediate internal succession, this keyword specifies the size of the largest program (load module) in the succession chain.

One need not consider resident load modules in this calculation, but keep in mind that load modules may be resident one session and not in the next session. For safety considerations, it might be prudent to always include all possible load modules in the calculation.

The MAXsize= specification is also required for TIP/30 programs which call the TIPJUMP subroutine.

MCSsize=n

The size of the MCS area (TIP/30 screen I/O area) for TIP/30 programs.

OUTsize=n

The Output Message Area (OMA) size if this is a program run under IMS Emulation.

PRIority=n

Override the default execution priority for this program.

By default, foreground or background programs run at the scheduling priority specified in the TIP/30 job control or TIPGEN parameters.

This keyword may be used in the TIP/30 catalogue entry to force a specific execution priority for this program.

The value specified is relative to 1 plus the base execution priority of TIP/30. For example, if TIP/30 was executed at priority 1 (often the case) specifying PRI=3 would run this program at 1+1+3=5.

If the value specified by this keyword exceeds the maximum priority level available, the value is forced to the lowest (numerically highest) value permitted.

S34=

Specifying S34=YES indicates that this program is a TIP/30 program written in RPG II language and is to behave as an OS/3 work station RPG program or a System/34 RPG online program.

The RPG interface routines supplied with TIP/30 simulate the ability of function keys to return data as well as setting the appropriate "K" indicator.

S34=YES also implies that the program is not controlling screen I/O using the features provided by the MCS interface packet.

SECurity=

The security level assigned to this program.

May be specified as a number between 1 and 255 (inclusive).

May be specified as one of: (TECH, MAST, SYST, PROG, APPL) representing (respectively) security level (1, 9, 19, 29, 32).

To access this program, a user must be at a security level numerically less than or equal to this value and must be a member of this program's group!

Default is SEC=PROG.

SHRDsize=n

This is an alternative spelling of the VOLatile= keyword (see description following).

SuBPRoG=

YES indicates that this program entry applies to the definition of a TIP/30 or IMS subprogram.

SUBPROG=YES must be specified to allow the specification of a REUSABLE TIP/30 subprogram (TIP/30 programs may not be reusable UNLESS they are subprograms).

TiMeLocK=(bgn,end)

The hours of the day that this program is NOT available.

Specified using a 24-hour clock.

Eg: TIMELOCK=(0830,1730) or TIMELOCK=(1800,0900).

Default: program is NOT time locked.

To rescind time locking specify TIMELOCK=NO.

Note: *During the TIMELOCK period, the program may not be entered; if it began execution before the TIMELOCK period it is allowed to continue (it is NOT aborted!).*

TPmode=

YES implies that this program normally does not execute for any significant length of time (the WMI utility is a good example). Knowledge of this behaviour enables TIP/30 to schedule the program into paged memory more or less in the first available place.

The scheduler will not use the usual complex algorithm to decide where (in memory) to load this program since it allegedly will not run very long.

Default: NO (use "normal" scheduling algorithm).

Specifying YES for an interactive program (one that DOES carry on extended conversations with the terminal) might result in the TIP/30 scheduler placing this program in an area of memory that is not the "best" place for the program.

TRANslat=YES

For input messages received by TIPTERM or IMS emulation all alphabetic characters are forced to upper case.

This specification is not necessary if the program uses the TIP/30 Message Control System (screen formats) since upper case translation may be controlled on a field-by-field basis with a screen format.

Default=NO.

TYPE=

IMS indicates that this is an IMS program (this could have been specified by IMS=YES).

TIP indicates that this is a TIP/30 program

CICS indicates that this is an IBM CICS program that is to be run using the CICS emulation package ("CAPIR" — from Chaparral Inc.).

Default: TIP/30 program.

USeage=

Define the level of reentrancy of the load module. This keyword is not misspelled; since lower case letters are optional, the spelling "USeage=" permits the keyword to be abbreviated as "USE=" or "USAGE=" etc.

REENT This program is reentrant.

This may be specified for sharable COBOL programs (those which compiled without diagnostics with the shared code option — OUT=(M) for COBOL68, IMSCOD=YES for COBOL74).

If the program is a sharable COBOL program the shared code VOLATILE data area size must also be specified (see VOLATILE= keyword description).

If a COBOL program was compiled using the IMSCOD=REN compiler option, the COBOL program is treated as a reentrant program; otherwise, it is considered SHARABLE (TIP/30 must reestablish each user's VOL area during a process switch — this is transparent to the programmer but does represent system overhead).

REUSE The program is serially re-useable. Only one process is allowed to use this program at a time. The process must terminate before the load module may be used by another process.

This specification is only valid for IMS programs running under emulation or TIP/30 subprograms which are not reentrant.

For sharable COBOL programs the VOLATILE data area size (VOLATILE=) must also be specified.

RELOAD The load module is to be reloaded each time the program is used.

This entry is required if the program is neither reentrant nor reusable.

VOLatile=n

The size (in bytes) of the shared code volatile data area of a sharable COBOL program.

Maximum value supported by TIP/30: 4096 bytes.

The size of the shared code volatile data area is reported by the COBOL-68 compiler on the last page of the compilation listing (near the diagnostics summary).

The size of the shared code volatile data area is reported by the COBOL-74 compiler on the first page of the compilation listing (compilation summary).

If this program calls any data base management routines, add four times the number of parameters in the longest "CALL" parameter list to the size reported by the COBOL compiler.

If the program calls DMS, it is necessary to add 256 bytes to the VOL size reported by the COBOL compiler (if the program is reentrant COBOL, specify 256 bytes).

WoRKsize=n

The size of the work area required by this program.

For COBOL-74 programs that were compiled with the IMSCOD=REN option, the value specified for the work area must be the size of the work area plus the amount reported by the compiler for the reentrant control area.

If the program calls linked COBOL-74 subroutines, you must include the sum of all of the reentrant control areas for the subroutines too.

XMIT=

The terminal control page XMIT setting required by this program.

Choices are: VAR, CHAN or ALL.

If this keyword is specified TIP/30 alters the terminal control page to the specified value before entering the program.

Default: no control page modification is performed.

. (period) The last keyword specified as part of the Prog command should be followed by a period (to indicate that the command is finished).

Example of dialogue defining a PROGRAM:

```
TIP?▶CAT
CAT(1)?▶PROG EDP/TESTPROG SECUR=PROG LOADM=TEST10 CDA=256
Continue PROG statement▶ WORK=1024 USAGE=RELOAD DEBUG=YES
Continue PROG statement▶ MCS=2048 FILES=PAYMAST,PAYDETL.
 PROG catalogue record added  : EDP/TESTPROG
CAT(1)?▶
```

This example illustrates the addition of a program entry in the TIP/30 catalogue for a program with a transaction code "TESTPROG".

This program definition is applicable to users who are members of group "EDP" and have security of (at least) programmer level (numerically less than or equal to 29).

The load module is named "TEST1000" (note that trailing zeroes are automatically supplied by CAT) and this load module requires a CDA of 256 bytes, a work area of 1024 bytes and an MCS area of 2048 bytes.

The load module is used in a "reload before use" fashion; the program IS to be run with hardware storage protection and the files with logical file names "PAYMAST" and "PAYDETL" are to be accessed on behalf of the program (the program does not explicitly call the TIP/30 file system to "OPEN" the files).

3.11.8. Hints for Program Testing

When testing a new program it is recommended that the transaction be catalogued with somewhat larger areas (CDA, MCS, WORK, IMA, OMA, VOL) than actually required.

When the program is completely tested the TIP/30 catalogue should be updated to reflect the correct sizes.

During program development the areas tend to grow and the programmer often forgets to keep the catalogue information up to date.

WARNING

Declaring too small an area size in the TIP/30 catalogue may result in some portion of another program (or TIP/30) being overwritten and may cause transactions or TIP/30 to terminate abnormally.

Catalogue the program being tested as USAGE=RELOAD and DEBUG=YES. When testing is complete change the usage to reflect the program's attributes and (if desired) remove debug options.

If the transaction program is compiled by the OS/3 COBOL-85 compiler, TIP/30 validates the sizes of the LINKAGE SECTION level 01 areas as defined to the compiler. The values specified in the TIP/30 Catalogue entry for the transaction must be greater than or equal to the amount calculated by the compiler. If the area sizes are not large enough, TIP/30 aborts the program before execution of the transaction begins.

3.11.9. FILE — Cataloguing a File

The FILE catalogue manager command creates or updates a file entry in the TIP/30 catalogue.

All online files must be catalogued in the TIP/30 catalogue — libraries and Data Management files.

A further recommendation is that the user take advantage of the logical naming of files and the assignment of security codes as these capabilities enhance the flexibility and security of the system.

Syntax:

```
File  group/LFN  kwd= kwd= ...
```

Where:

group Positional parameter used to indicate that the definition of this file applies to users who are members of this group.

lfn A required positional parameter identifying the logical file name which online programs must use to access this file.

This may be the same as the LFD name, BUT it need not be the same. For example, a user can create a file entry for a logical file name "J" that in fact refers to the file with LFD=YJCS.

The following table summarizes the keywords recognized by the FILE command:

Table 3-6. FILE Definition Keywords

Keyword	Description
DMCL=	Define DMS DMCL name.
KeyREF=	Default index to use when none specified.
LaBeL=	Physical LBL name (library files).
LFD=	LFD name of the file as given in JCL.
ReaD=	"NO" — read not allowed.
ReadPWD=	READ password.
SECurity=	Security level required to use this file.
VSN=	Volume serial number of disk volume.
WRITE=	"NO" — write not allowed.
WritePWD=	WRITE password.
(period)	Signal end of keywords for File command.

DMCL= This keyword indicates that the logical file name (lfn) is really a logical DMCL name (for DMS use) and it is to refer to the actual DMCL name as specified by this keyword.

DMCL= and LFD= are mutually exclusive keywords.

KeyREF= For MIRAM files, the default index to use (1-5) if a program chooses to not supply an index specification explicitly when using the TIP/30 file system.

This keyword overrides the specification of the default key of reference for the file.

Default: TIPGEN PKEY= specification for this file.

LaBeL= LBL name of a library that is catalogued in the OS/3 system file catalogue (YCAT) that is being dynamically accessed.

See section following on accessing OS/3 libraries.

If the LBL name contains an imbedded space or period character or is longer than eight characters, specify the LBL name in quotes. For example:

```
LBL='PAYROLL SOURCE'
```

LFD= LFD name of the file as given in the TIP/30 JCL and in the TIP/30 generation.

If this file entry is for a dynamically accessed OS/3 library, be sure to specify an LFD name that does not conflict with an LFD name that is used to reference another file — the LFD name is not so important for dynamically accessed libraries, but accidently using an LFD name of a different file can cause unpredictable results. See also "3.11.12. Accessing OS/3 Libraries" on page 3-40.

ReaD= READ=NO indicates read operations not permitted for this file — output only file for this group.

Default: whatever was specified for the LFD in the TIP/30 generation parameters or TIP/30 job control.

ReadPWD=

Read password for a library which is catalogued in the OS/3 catalogue (YCAT) with a read password.

Specifying the password in the TIP/30 catalogue allows you to depend on the TIP/30 security system; users who have legitimate access to this file never need to explicitly specify the password.

Default: no read password is used.

SECurity=

The security level required to access this file.

May be specified as a number between 1 and 255 (inclusive).

May be specified as one of: (TECH, MAST, SYST, PROG, APPL) representing (respectively) security level (1, 9, 19, 29, 32).

To access this file, a user must be at a security level numerically less than or equal to this value and must be a member of this file's group!

Default is SEC=PROG.

VSN= For an OS/3 library that is being dynamically accessed (and has NOT been catalogued in YCAT), the disk volume serial number where the library can be found must be specified via this keyword (since it cannot be dynamically obtained from YCAT).

Also see "3.11.12. Accessing OS/3 Libraries" on page 3-40.

WRITE= WRITE=NO indicates write operations not permitted for this file — input only file for this group.

Default: whatever was specified for the file in the TIP/30 generation parameters or TIP/30 job control.

WritePWD=

Write password for a password protected OS/3 library that is to be dynamically accessed.

Specifying the password in the TIP/30 catalogue allows you to depend on the TIP/30 security system; users who have legitimate access to this file never need to explicitly specify the password.

Default: no write password is used.

. (period) The last keyword specified as part of the File command should be followed by a period (to indicate that the command is finished).

Example of dialogue defining a FILE:

```
TIP?▶CAT
CAT(1)?▶FILE EDP/JCS LFD=$Y$JCS SECUR=PROG.
 FILE catalogue record added  : EDP/JCS
CAT(1)?▶
```

This example illustrates the definition of a file (a library) with the logical name "JCS". The file is actually referencing the file with the LFD "YJCS".

This TIP/30 Catalogue entry implies that users with at least programmer level security and membership in the group "EDP" are able to access this file by referring to the (logical) file name JCS.

3.11.10. GROUPSET — Defining a Groupset

The CAT manager **Groupset** command may be used to define a groupset or to update an existing groupset record in the TIP/30 catalogue.

If the named groupset already is defined in the catalogue, the "G" command is considered an update rather than an "add" operation.

Syntax:

```
Groupset  name   kwd= kwd=
```

Where:

name The name of the groupset. This positional parameter is required.

The following table summarizes the keywords recognized by the GROUPSET command:

Table 3-7. GROUPSET Definition Keywords

Keyword	Description
SECurity=	Security for groupset record.
GRouPSet=	A list of up to 16 groups in this set.
NextGRPs=	Point to next GROUPSET record in chain.
(period)	Indicate end of Groupset command.

SECurity=

The security level for this groupset record.

GRouPSet=

A list of up to 16 group names (with or without an associated individual security level) that are part of this defined set of group names.

This keyword may be repeated to facilitate keyboard entry.

If a security level is not explicitly coded with a group name, the security level associated with that group name defaults to the security of the user accessing the group information.

NextGRPs=

The name of the next groupset record to link in the chain of groupset records.

This keyword allows you to effectively connect an arbitrarily large number of groupset records in a chain.

Note that the name supplied with this parameter is not validated when it is entered in the TIP/30 catalogue - if the name is not valid at the time it is used it is ignored and the chaining of groupset records terminates.

. (period) The last keyword specified with the Groupset command should be terminated with a period to indicate that the command has ended.

Example:

```
TIP?▶CAT
CAT(1)?▶GRPS ALL SECUR=PROG GRPS=EDP GRPS=ACCTNG,19
CAT(1)?▶       GRPS=ENG,65,PAYROLL,70.
CAT(1)?▶ GRPS catalogue record added  : ALL
CAT(1)?▶
```

This example creates a groupset named "ALL". Any user who is assigned membership in the groupset "ALL" (by specifying GROUPSET=ALL in their user record) have additional elective groups in their order of search: EDP, ACCTNG, ENG and/or PAYROLL (with the indicated security levels within those groups).

3.11.11. Catalogue Statement Continuation

The CAT program processes only the first 72 characters of an input line.

When entering data on the terminal, type up to 72 characters and then press XMIT (it is wise to completely finish a keyword before continuation becomes necessary).

The CAT program prompts again indicating that a continuation is expected. Leave at least one space after the SOE character in the prompt and continue to enter additional keyword parameters.

When entering the last line of a CAT command, terminate the line with a period (this signals CAT that continuation is NOT required!).

CAT automatically terminates the previous command if it reads an input line that has a command character as the first character.

3.11.12. Accessing OS/3 Libraries

OS/3 libraries need not be specified in the TIP/30 generation. They do not even have to be defined in the TIP/30 Job Control stream.

An OS/3 library that is not explicitly generated in the TIP/30 system may be dynamically accessible by TIP/30 subject to the following considerations:

- the disk VOLUME must be online
- the JCL for the library need not be present in the Job Control stream for TIP/30
- the (library) file MUST be specified in the TIP/30 catalogue.

Since the library may or may not be defined in the TIP/30 Job Control and the library may or may not be catalogued in the OS/3 system file catalogue (YCAT), the following rules are enforced by TIP/30 when searching for access to a library:

First choice

The location of the library LFD according to any specifications in the TIP/30 Job Control stream.

Second choice

Search the OS/3 system file catalogue (YCAT) for the LBL= specified in the TIP/30 catalogue entry for the library.

Third choice

Search the vtoc for the file using the information supplied in the TIP/30 catalogue FILE statement keywords LBL= and VSN=.

Dynamically accessed libraries should be included in the TIP/30 job control (even though they may be defined in YCAT) to permit automatic secondary allocation to take place when necessary.

3.11.13. Updating Catalogue Records

To update an existing catalogue record one need only specify sufficient information to identify the key of the catalogue record desired and supply (via keywords) the information which is to be changed or added.

Use the appropriate CAT command (User, File, Prog, Groupset) and be sure to supply the requisite parts of the key:

Example:

```
TIP?▶CAT
CAT(1)?▶USER  TOMMY  PROG=MENU.
CAT(1)?▶ USER catalogue record updated: TOMMY
CAT(1)?▶PROG  AP/UPDT  CDA=768.
CAT(1)?▶ PROG catalogue record updated: AP/UPDT
CAT(1)?▶FILE  AP/MAST  SECUR=88.
CAT(1)?▶ FILE catalogue record updated: AP/MAST
CAT(1)?▶
```

Note: *The CAT program LIST command displays the information for a catalogue record in a manner which is suitable for screen cannibalization and re-entry.*

3.11.14. Updating by Load Module Name

Often it is necessary to change information that is related to the load module (ie: WORKsize CDAsize USAGE etc). In this situation it is tedious to make the same change to a (possibly) large number of transaction entries that reference that load module.

The CAT program allows "mass" updates by load module name.

Syntax:

```
Prog  grp/trid  LDM=(oldldm [,newldm])  kwd= kwd= ...
```

Where:

grp A specification of the groups that are to be searched.

This entry is required and may be specified using standard prefix notation.

trid A specification of the transaction names that are to be considered.

This entry is required and may be specified using standard prefix notation.

oldldm The first sub-parameter of the LDM= keyword specifies the existing load module name.

newldm The second sub-parameter of the LDM= keyword may be used to specify a new load module name (since LDM= is used to locate candidates for change).

kwd= The other keywords that are valid for the PROG statement may be specified here to alter the information for any PROG entry selected by the load module name search.

Example:

```
TIP?▶CAT
CAT(1)?▶PROG */* LDM=(PAY020,PAYX20) WORK=4800.
```

This example indicates that all transactions in all groups are to be searched for entries that have an existing load module specified as "PAY020". The load module name is to be changed to "PAYX20" and the worksize is to be changed to 4800 bytes.

Additional Considerations:

An important point to realize is that normal security rules apply during this "extended" search — that is, a user can only access entries that he is entitled to access.

3.11.15. Listing Catalogue Entries

The list command allows the user to display (at the terminal) the information stored in one (or more) catalogue entries.

The listing is produced in physical catalogue order (undefined order!) unless the LS variant is used (LS means "list sorted").

Syntax:

```
L[S] Grp/Id/Elt [,Type] [ GRouP=xxx ]
                        [ DATe=yymmdd ]
                        [ DATe<=yymmdd ]
                        [ LFD=xxxxxxxx ]
                        [ LoaDM=xxxxxxxx ]
```

Where:

Grp Userid or group name of the catalogue records to be listed.

This value must be a group to which the user belongs unless the user has a security level of Tech or Master.

If this parameter is not given, the user's private group (userid) is assumed.

Id Name of the item (program or file) to be displayed.

If not given, all items in the specified group are displayed.

Elt The element name of a dynamic file. This parameter is only valid when used with dynamic file entries.

If not given, all elements are processed.

Type The type(s) of catalogue records to list. The value for this parameter may be as follows:

* — All record types

P — Program records

U — Userid records

F — File (all files) records

— D = Dynamic file records

— E = Edit file records

— S = System file records

G = Groupset records

More than one type may be specified by concatenating type codes (for example, "DE" means dynamic or edit type).

If type is omitted, all types are processed.

Note: for the first three positional parameters (Grp, Id, and Elt), the value may be specified using standard prefix notation.

GRouP= This optional keyword may be used to limit the scope of the command to those catalogue entries in this group.

Prefix notation may be used for this value.

DATe= This optional keyword may be used to limit the scope of the command to those catalogue entries which have a "last opened" date EQUAL TO this date.

This keyword is only applied to Dynamic or Edit buffer entries.

Prefix notation may not be used for this value.

DATe<= This optional keyword may be used to limit the scope of the command to those catalogue entries which have a "last opened" date LESS THAN OR EQUAL TO this date.

This keyword is only applied to Dynamic or Edit buffer entries.

Prefix notation may not be used for this value.

LoaDM= This optional keyword may be used to limit the scope of the command to those catalogue entries which have a load module name that matches this value.

This keyword is only applied to PROG entries.

Prefix notation may be used for this value.

LFD= This optional keyword may be used to limit the scope of the command to catalogue entries with an LFD name that matches this value.

This keyword is only applicable to standard system files.

Prefix notation may not be used for this value.

Example:

```
TIP?▶CAT
CAT(1)?▶L *,*pay
```

This command lists all catalogue records (since no type is given) of any group (* alone means "match any") and of any name that starts with the letters "PAY".

3.11.16. Listing Dynamic File Entries

An abbreviated version of the List command is available to simplify listing catalogue entries for dynamic files.

Syntax:

```
LD   grp  [,id] [,elt]  [keyword qualifiers]
```

Where:

grp	The dynamic file group name to select. This parameter may be specified using standard prefix notation. If omitted, the user's private group (userid) is assumed.
id	The file name of the dynamic files to list. This parameter may be specified using standard prefix notation. If omitted a value of * is assumed (any id).
elt	The element name of the dynamic files to list. This parameter may be specified using standard prefix notation. If omitted a value of * is assumed (any id).
qualifiers	Any of the standard CAT list keyword qualifiers (GRouP=, DATe=, DATe<=, LFD= or LoaDM=) as described in page 3-43.

Example:

```
TIP?▶CAT
CAT(1)?▶LD EDP
```

This command lists all dynamic files in the group "EDP".

3.11.17. Listing Edit Buffer Entries

An abbreviated version of the List command is available to simplify listing catalogue entries for edit buffers.

Syntax:

```
LE   grp  [,name]        [keyword qualifiers]
```

Where:

grp The edit buffer group name to select.

This parameter may be specified using standard prefix notation.

If omitted, the user's private group (userid) is assumed.

name The name of the edit buffer to be listed.

This parameter may be specified using standard prefix notation.

If omitted, all edit buffers in the selected groups are listed.

qualifiers Any of the standard CAT list keyword qualifiers (GRouP=, DATe=, DATe<=, LFD= or LoaDM=) as described in page 3-43.

Example:

```
TIP?▶CAT
CAT(1)?▶LE EDP
```

This command lists all edit buffers in the group "EDP".

3.11.18. Listing File Entries

An abbreviated version of the List command is available to simplify listing catalogue entries for Data Management (standard system) files.

Syntax:

```
LF   grp  [,name]        [keyword qualifiers]
```

Where:

grp The group to which the desired file entry pertains.

This parameter may be specified using standard prefix notation.

If omitted, the user's private group (userid) is assumed.

name The logical file name of the file entry to list.

This parameter may be specified using standard prefix notation.

If omitted, a value of "*" is assumed (all logical file names).

qualifiers Any of the standard CAT list keyword qualifiers (GRouP=, DATe=, DATe<=, LFD= or LoaDM=) as described in page 3-43.

Example:

```
TIP?▶CAT
CAT(1)?▶LF EDP
```

This command lists all files (of any type) that are defined in the group "EDP".

3.11.19. Listing Program Entries

An abbreviated version of the List command is available to simplify listing catalogue entries for PROGRAM entries in the TIP/30 catalogue.

Syntax:

```
LP   grp  [,name]       [keyword qualifiers]
```

Where:

grp The group to which the desired PROG entry pertains.

This parameter may be specified using standard prefix notation.

If omitted, the user's private group (userid) is assumed.

name The name of the program entry to list.

This parameter may be specified using standard prefix notation.

If omitted, a value of "*" is assumed (all program names)

qualifiers Any of the standard CAT list keyword qualifiers (GRouP=, DATe=, DATe<=, LFD= or LoaDM=) as described in page 3-43.

Example:

```
TIP?▶CAT
CAT(1)?▶LP EDP *
```

This command lists all programs defined in the group "EDP".

3.11.20. Listing User Entries

An abbreviated version of the List command is available to simplify listing catalogue user records.

Syntax:

```
LU   userid
```

Where:

userid The userid of the record to be listed.

This parameter may be specified using standard prefix notation.

Note that only TECH or MAST level users may list userid records other than their own.

Example:

```
TIP?▶CAT
CAT(1)?▶LU FRED
```

This command lists the userid record for user FRED.

3.11.21. Deleting Catalogue Entries

Entries in the catalogue may be deleted by using the delete command. The delete command allows the user to delete entries by prefix specification but issues a warning prompt if the number of entries to be deleted exceeds 10 — one final chance to confirm.

A fine rule of thumb is to List the entries to be deleted and use exactly the same parameters for the DELete command that were used to list the entries (the F1 function key can be used to recall the list command).

A user can only delete those catalogue entries to which he has access.

The CAT program does not allow a user to delete his own userid record.

Syntax:

```
DELete  Grp,Id,Elt,type [keyword qualifiers]
```

Where:

All four parts of the key must be specified.

Grp Userid or group name of the catalogue records to be deleted.

This value must be a group to which the user belongs unless the user has a security level of Tech or Master.

If this parameter is not given, then the user's private group (userid) is assumed.

Id Name of the item (program or file) to be deleted.

If not given, then all items in the specified group are deleted.

Elt The element name of a dynamic file. This parameter is only valid when used with dynamic file entries.

If not given, all elements are processed.

type The type of catalogue records to delete. The value for this parameter may be as follows:

More than one type may be specified by concatenating the alphabetic type codes (for example, "DE" means dynamic or edit type).

qualifiers Any of the standard CAT list keyword qualifiers (GRouP=, DATe=, DATe<=, LFD= or LoaDM=) as described in page 3-43.

Additional Considerations:

When a request is made to delete a user record, the CAT program checks whether that user owns any TIP/30 Dynamic Files. If dynamic files exist, the CAT program prompts to confirm that the user is to be deleted.

If the response to the prompt is "YES", the user record and the dynamic file or files are deleted.

If the response to the prompt is "NO", the delete command is not performed (the user record remains in the catalogue).

3.11.22. Recall Last Command

The CAT program recognizes F1 (Function key 1) as a valid command.

Pressing the F1 key causes the CAT program to output the last command that was entered from the terminal.

The text of that command may then be altered (on the screen) and transmitted — a popular trick is to recall a List command and alter it slightly so that it is a DELete command with exactly the same parameters — "what you listed is what you delete").

Example:

```
TIP?▶CAT
CAT(1)?▶L EDP,*,,E
CAT(1)?▶  ...      { a list of edit buffers in the  }
CAT(1)?▶  ...      { group EDP now fills the screen }
CAT(1)?▶F1
▶L EDP,*,,E        { CAT outputs last command entered }
CAT(1)?▶
```

3.11.23. Writing Catalogue Entries

It is often necessary to make "bulk" changes to the TIP/30 catalogue. A WRITE command is recognized by the CAT program to make the process of mass changes somewhat easier to accomplish.

The write command creates a library element containing the keyword information for selected catalogue entries. The information is written in the format that the CAT program expects the information to be input.

Once such an element is created, the user may use a standard Editor (FSE for example) to make massive changes to the information.

The resultant (changed) element of information could then be supplied as input to the CAT program to effect all the changes. See the example following for details.

The Write command processes PROGRAM and FILE entries (standard OS/3 file entries) ONLY. User records, edit buffer records, and dynamic file records are ignored by the WRITE command.

The Write command ignores catalogue entries for PROGRAMS that are supplied with TIP/30 (it does this by ignoring programs which have a load module name prefixed by "TT$", "TT-", "TT@", "QT$" or "XT$").

The alternative spelling of the write command ("WA") may be used to force the WRITE command to process ALL catalogue records ("WA" means "write all").

Syntax:

```
W      Grp,Id,Elt,type [,lib [ ,elt]  [keyword qualifiers]
WA
```

The parameters of the Write command are identical to the parameters required by the LIST command (see previous description of that command for details) with the following exceptions:

lib The logical file name of the library used to output the catalogue information.

If omitted, the library name "RUN" is assumed.

elt The element name created in the output library.

If omitted, the name "LISTCAT" is assumed.

Example:

```
TIP?▶CAT
CAT(1)?▶w edp,*,,p sysgen/catsmash
```

This example writes the catalogue information for all programs in the group EDP to a library element named CATSMASH in library SYSGEN.

The user can then invoke a standard editor (like FSE) and make changes, additions or deletions to the information in SYSGEN/CATSMASH as desired.

When all changes are finished, the user can submit that element as input to the CAT program by issuing the following command line:

```
TIP?▶CAT <SYSGEN/CATSMASH
```

The less than symbol "<" indicates to the TIP/30 command line processor that the input for the program is to be taken one line at a time (as requested by the CAT program) from the specified library element instead of soliciting the user at the terminal. See description of "Input redirection" in the documentation of the Program Control System (PCS).

3.12. CCA — ICAM Statistics Display

CCA is a TIP/30 utility program which displays information and statistics derived from tables maintained by ICAM (the OS/3 communications control program).

ICAM must be generated with "STAT=YES" on the BUFFER statement and "STATS=YES" on LINE statements (note the different keyword spelling).

This specification causes ICAM to collect statistics for lines and buffers. The CCA program can assist the systems programmer to determine the source of communications problems and to evaluate the performance of ICAM.

The CCA program displays the statistics maintained by ICAM using TIP/30 screen formats. The information may (optionally) be printed for later analysis.

ICAM statistics are accumulated (by ICAM) from the time ICAM is initialized. ICAM shutdown may not correspond to a TIP/30 shutdown (in a Global ICAM environment).

Syntax:

① `CCA`

② `CCA command [,name]`

Where:

Syntax format ① is used to start the CCA program in interactive mode. CCA prompts the user with a screen format and permits the user to execute a number of commands. Format ② is used to execute a single CCA command and then terminate.

The commands recognized by the CCA program are:

Table 3-8. CCA Commands

Command	Function
A	Display A.R.P. usage.
B	Display buffer usage.
G	Display Locap statistics (Global ICAM only).
K	Display Linkpak statistics (DCA ICAM only).
E	End execution of CCA program.
L	Display line information and statistics.
O	Display Opcom statistics.
P	Print all available statistics.
Q	End execution of CCA program and logoff TIP/30.
T	Display terminal information and statistics.
U	Display Uduct statistics (DCA ICAM only).

The optional second parameter is a starting LINE/LOCAP/TERM name which may be used to cause the selected command to begin its display at the LINE, LOCAP, or TERM specified.

If the CCA program is invoked with an explicit command, CCA performs that command and then terminates (most commands require the user to press the MSG WAIT key to exit the command).

If no command is present on the command line, CCA displays the following screen format to simplify the entry of commands and assumes that the user wishes to run the CCA program interactively (that is, issue one or more commands):

```
TF$CCA0A        * *   I C A M   C 2   S T A T I S T I C S   * *  11:22  89/07/26

                              CCA Information
                      Network    - NET2 Type       - GBL
                      Lines      -   4  Terminals -  22

                         Select one of the following:

                         A   -   Arp statistics
                         B   -   Network buffer statistics
                         G   -   Locap statistics    (GBL ICAM only)
                         K   -   Linkpak statistics (DCA ICAM only)
                         L   -   Line statistics
                         O   -   Opcom statistics
                         T   -   Terminal statistics
                         U   -   Uduct statistics    (DCA ICAM only)
                         P   -   Print all of the above
                         E   -   End program
                         Q   -   End program and logoff

                         Enter selection here  --> _
                         Enter Locap/Line/Term --> ____
                   'TIP/30 CCA Inquiry' - Version = 4.0 (89/06/20)
F1/5:Redisplay,  F2/6:Next CCA,  F3/7:First CCA,  F4/8:Quit,  Msg-Wait:End
```

Note that information about the first (and possibly only) CCA (Communications Control Area) is displayed at the top of the display.

The network password (if one exists) is displayed only if the user of the CCA program is a Tech or Master level TIP/30 user.

The user may enter a command (the list of available commands is displayed) or may press one of the available function keys.

Where:

selection One of the one character commands as shown.

The standard "E" and "Q" commands are not described in further detail since they are fairly obvious.

LOCAP/LINE/TERM

An optional LOCAP or LINE or TERM name to indicate where to begin a command's display.

Some of the commands display information about LOCAPs, LINES, or TERMINALs. This field may be used to indicate that such a display is to start with the indicated entry.

This is helpful when the terminal that interests you most is the 114th terminal in the CCA and you don't have the time to scroll through the first 113 terminals.

This field is interrogated by the "Global", "Line", and "Term" commands (only).

F1/F5 Refresh the display (retransmit the display).

F2/F6 Move forward to the next CCA (if more are defined).

More than one CCA may be defined in an ICAM network; CCA reports statistics on all CCAs (provided the CCA was generated with the correct keywords for statistics accumulation).

F3/F7 Display the first CCA in the network.

This is often used to reset the display to select the first CCA in the network after examining other CCA information.

F4/F8 Terminate the CCA program and log off TIP/30.

MSG WAIT End the CCA program.

Additional Considerations:

The usual mode of operation is to use F2 to select the correct CCA to examine (if there are multiple CCAs in the ICAM) and then proceed to issue commands to display information about that CCA.

To move to a different CCA after displaying information about another CCA, the user must return to this main display and use the function keys to select another CCA.

3.12.1. A — ARP Utilization

Activity Request Packets (ARP) are ICAM internal table entries that are used by ICAM when scheduling network functions. There are a fixed number of these packets generated into the CCA (a formula for predicting the number needed can be found in the appropriate ICAM publication for your type of network).

CCA displays the usage of the available ARPs. The usage is displayed in an inverted fashion: the number of times a certain number of ARPs was available is shown.

In effect, the usage is displayed as the penetration into the available pool of ARPs.

```
TF$CCA1A        * *   I C A M   M 1   S T A T I S T I C S   * *   13:46  89/01/09

                      A.R.P. information for NET1 network

Number: 90  Available: 74  Thresh: 10  Size:   56  Defers:     0  Rejects:     0

                           Penetration information

       Remaining    Times    Remaining    Times    Remaining    Times

           89           1        79           2        69         132
           88           2        78           1        68          20
           87           1        77           1        67           9
           86           1        76          17        66           3
           85           1        75          23        65           1
           84           2        74          19        64           0
           83           2        73         165        63           0
           82           1        72         750        62           0
           81           1        71         695        61           0
           80           1        70         372        60           0

         F1/5:Redisplay,    F2/6:Next screen,  F3/7:First screen
         F9:Refresh stats,  F13:Quit,          F15:End,  Msg-wait:Menu
```

In this example the error field (not shown) would be blinking with the message "More A.R.P.S to display." because all of the A.R.P. information cannot be shown on a single display.

The above example illustrates that there were 0 times when there were (only) 64 ARPs available and there was as few as 65 ARPs available only 1 time.

From such information, the system programmer is able to determine whether or not there is an abundance of ARPs generated into ICAM.

3.12.2. B — Buffer Utilization

ICAM network buffers (or simply buffers) are internal areas used by ICAM to hold data and control characters that are being sent to a terminal or being sent by a terminal.

When the system programmer generates ICAM, he specifies the number and size of these buffers.

It is very important to strike a reasonable balance between the number of network buffers and their size. The size is related to the average size of input or output messages (the TIP/30 utility program STATUS can be used to determine the average input and output message lengths).

The buffer display in the CCA program displays the usage of ICAM buffers in the same format as is used for the ARP penetration display.

The number of times a certain number of buffers were available is shown.

```
TF$CCA1A       * *   I C A M   M 1   S T A T I S T I C S   * *   13:46  89/01/09

                     Buffer information for NET1 network

Number: 35  Available: 35  Thresh:  5  Size:  256  Defers:      0  Rejects:      0

                            Penetration information

       Remaining     Times     Remaining     Times     Remaining     Times

           34          386         24            0         14            0
           33          173         23            0         13            0
           32          127         22            0         12            0
           31          102         21            0         11            0
           30           74         20            0         10            0
           29           58         19            0          9            0
           28           51         18            0          8            0
           27           47         17            0          7            0
           26           12         16            0          6            0
           25            1         15            0          5            0

         F1/5:Redisplay,    F2/6:Next screen,  F3/7:First screen
         F9:Refresh stats,  F13:Quit,          F15:End,  Msg-wait:Menu
```

In this example, the error field (not shown) would also be blinking with the message "More Buffers to display." because all of the buffer information cannot be shown on a single display.

The above example illustrates that there were 0 times when there was (only) 24 buffers available and there was as few as 25 buffers available only 1 time.

If this information was the result of an average period of network activity, one could conclude that about 15 buffers are wasted memory (it is important to realize that such a conclusion should be based on several observations — not an isolated incident).

3.12.3. G — LOCAP Information

The CCA "G" command displays information about the LOCAPs that are generated in a Global ICAM network. LOCAPs exist only in a Global ICAM environment.

Each LOCAP is displayed by name along with information that ICAM maintains for each. The appropriate ICAM manual may be useful for establishing the meaning of the information displayed.

The Global command is generally used to find out what LOCAPs are configured and whether they are available (up).

```
TF$CCA4A      * *  I C A M  M 1  S T A T I S T I C S  * *  13:46  89/01/09

                   Locap information for NET1 network


            Name    Type    Active    Node    Queues    Sessions

            TIP1    TCI       No      YYZ        0         0
            TIP2    TCI       Yes     YYZ        1         1
            TIP3    TCI       Yes     ORD        1         2
            ISSN    DMI       Yes     YYZ        3         1









            F1/5:Re-display,   F2/6:Next screen,  F3/7:First screen
            F9:Refresh stats,  F13:Quit,          F15:End,  Msg-wait:Menu
```

In the example above the error field (not shown) would not have any message (since we are using a small network for our example and there are no more LOCAPs to display).

The information shown is more or less self-explanatory. The user could utilize the function keys (as advertised) if there was more LOCAP information available.

The user can specify the starting point of the display by providing a LOCAP name on the initial command screen or command line.

3.12.4. K — Linkpak Information

The CCA "K" command displays information about the utilization of Linkpak areas. Linkpaks are configured only in a DCA ICAM.

```
TF$CCA1A        * *   I C A M   C 2   S T A T I S T I C S   * *   15:00  89/02/27

                    Lnkpak information for NET2 network

Number: 90  Available: 84  Thresh:  3  Size:3,072  Defers:      0  Rejects:      0

                          Penetration information
                          More LINKPAKs to display.
      Remaining     Times    Remaining     Times    Remaining     Times

          89             1        79         508        69             0
          88             1        78         129        68             0
          87             2        77          32        67             0
          86            18        76           9        66             0
          85            20        75           1        65             0
          84       115,789        74           1        64             0
          83       167,505        73           1        63             0
          82        66,086        72           0        62             0
          81        12,250        71           0        61             0
          80         2,182        70           0        60             0

        F1/5:Redisplay,    F2/6:Next screen,  F3/7:First screen _
        F9:Refresh stats,  F13:Quit,          F15:End,  Msg-wait:Menu
```

3.12.5. L — Line Information

Information concerning the communications lines that are used by a CCA may be displayed by the CCA program. Certain LINE statistics are available only for lines that are generated in ICAM with STATS=YES specified.

Each line in the CCA is shown by name with the type, speed, number of terminals on the line, line status (UP/DOWN), the number of messages (input and output) and reported errors.

The latter two values are available only if statistics were requested in the ICAM generation.

```
TF$CCA2A        * *   I C A M   M 2   S T A T I S T I C S   * *   13:40  89/02/20

                      Line information for NET2 network


 Name    Device Type       Speed    Terms   Status    Messages     Errors    Hit/Rate

 L311    Loc Work Stn                1     Up          13,088         11       0.1%
 LIN1    Uniscope          9,600     12    Up               0          0
 LIN2    Uniscope          9,600     10    Up               0          0
 VLN1    UDLC ABM         56,000      0    Up               0          0


    F1/5:Re-display,   F2/6:Next screen,  F3/7:First screen
    F9:Refresh stats,  F13:Quit,          F15:End,  Msg-wait:Menu
```

In the example above the error field (not shown) would not have any message (since we are using a small network for our example and there are no more lines to display).

The information shown is more or less self-explanatory. The user could utilize the function keys (as advertised) if there was more line information available.

Additional Considerations:

The user can specify the starting point of the display by providing a LINE name on the initial command screen or command line.

3.12.6. O — Opcom Utilization

The CCA "O" command displays information about the utilization of the ICAM resource "Opcom". These packets are used for console operator communication.

```
TF$CCA1A        * *   I C A M   C 2   S T A T I S T I C S   * *   15:00  89/02/27

                      Opcom  information for NET2 network

Number:  3  Available:  3  Thresh:  0  Size:   84  Defers:      0  Rejects:      0

                            Penetration information

       Remaining     Times     Remaining     Times     Remaining     Times

            2            2
            1            0
            0            0









          F1/5:Redisplay,     F2/6:Next screen,  F3/7:First screen
          F9:Refresh stats,   F13:Quit,          F15:End,  Msg-wait:Menu
```

3.12.7. P — Print Report

The CCA "P" command is available to generate a printed copy of the current ICAM statistics. The PRINT command prints a copy of the ICAM statistics that are available (some information is available only in a Global ICAM network or DCA ICAM). In effect, the Print command cycles through all the other commands, directing the output to the site printer PRNTR by default.

The print command recognizes an alternate printer destination that is placed in the field "LINE/TERM/LOCAP" on the main menu screen.

If this command is supplied on the command line, an alternate printer destination may be specified as command line parameter 2:

```
TIP?▶CCA P C:CCA
```

The CCA program generates the printed statistics report and terminates without requiring intervention from the terminal user.

One could easily incorporate this as a function of a background program that issues a TIPSUB to the CCA program (with "P" as CDA parameter 1) once every few hours for example.

3.12.8. T — Terminal Information

Information concerning the communications terminals that are used in a CCA may be displayed by the CCA program. Certain TERM statistics are available only for lines that are generated in ICAM with STATS=YES specified.

Each terminal on each line of the CCA is displayed with two lines of information per terminal.

```
TF$CCA3A        * *   I C A M   M 1   S T A T I S T I C S   * *   13:46  89/01/09

                      Terminal information for NET1 network


Line    Term    Size      Address  Poll interval      Status
                Msg-in    Msg-out  In-retransmit  Out-retransmit  Polls  No-traffic
L311    T311    24X80       03 11          1.0          Up
                  25          91            0           0          116        90
L312    T312    24X80       03 12          1.0          Down
                   0           0            0           0            0         0
L313    T313    24X80       03 13          1.0          Down
                   0           0            0           0            0         0
LN08    ARC1    24X80       21 51          1.0          Up
                   0           0            0           0       45,166    45,064
        ARC2    24X80       21 52          1.0          Up
                   0           0            0           0            0         0
        ARC3    24X80       21 53          1.0          Up
                 104         180            0           0          282       282

        F1/5:Re-display,   F2/6:Next screen,  F3/7:First,  F4/8:Next line _
        F9:Refresh stats,  F13:Quit,          F15:End,     Msg-wait:Menu
```

In the example above the error field (not shown) would have a blinking message if more terminal information was available.

The information shown is more or less self-explanatory. The user could utilize the function keys (as advertised) if there was more line information available.

The information for each terminal is contained on two lines of the display (the second line contains the count of messages input and output via the terminal, the number of input and output retransmits, the number of polls and no-traffic replies received).

Additional Considerations:

The user can specify the starting point of the display by providing a TERM name on the initial command screen or command line.

3.12.9. U — Uduct Utilization

The CCA "U" command displays information about the utilization of Uduct buffers. Uduct buffers are configured only in a DCA ICAM.

```
TF$CCA1A       * *   I C A M   C 2   S T A T I S T I C S   * *   15:01  89/02/27

                     Uduct  information for NET2 network

Number: 90  Available: 85  Thresh:  3  Size:  112  Defers:     0  Rejects:     0

                           Penetration information
                           More UDUCTs to display.
      Remaining     Times    Remaining     Times    Remaining     Times

          89        33,810       79           45       69            0
          88        53,898       78           13       68            0
          87        39,800       77            6       67            0
          86        42,675       76            1       66            0
          85        46,184       75            0       65            0
          84        25,024       74            0       64            0
          83         9,590       73            0       63            0
          82         3,402       72            0       62            0
          81           856       71            0       61            0
          80           180       70            0       60            0

        F1/5:Redisplay,    F2/6:Next screen,  F3/7:First screen
        F9:Refresh stats,  F13:Quit,          F15:End,  Msg-wait:Menu
```

3.13. CH — OS/3 CHANGE Command

The CH transaction implements the OS/3 "CHANGE" console operator command (change job's execution queue). The CH transaction is actually a clone of the more powerful SYM transaction. The SYM transaction program can be invoked using an alias name that is interpreted as a specific command. When the SYM program is invoked with a transaction name of "CH", the SYM program assumes that the OS/3 command is "CH".

The OS/3 CHANGE command syntax is documented in the operation guide for your system.

The CH transaction requires the same syntax as the console command. The command is passed exactly as entered to the OS/3 command processor for execution as if it had been entered at the system console.

Example:

```
CH COBCOMP3,H
```

Change job "COBCOMP3" to High queue.

3.14. COPY — COPY Utility

The COPY transaction is a clone transaction of the generalized librarian utility transaction TLIB (see "3.90. TLIB — Librarian Services" on page 3-325) .

The COPY transaction invokes the TLIB program. When the TLIB program observes that the transaction name is not TLIB, it uses the transaction name as the implied command.

The command line options and parameters that are supplied with the COPY transaction code are interpreted by TLIB as parameters to the TLIB COPY command.

The end result is the ability to use COPY as an apparently stand-alone transaction.

Syntax:

```
COPY[,options]  parameters
```

Where:

options Any command line options (as recognized by TLIB) that pertain to the COPY command. See description of TLIB options.

parameters Parameters required by the COPY command of the TLIB program.

Example:

```
COPY,X  C:BUDGET.WKS SRC/BUDGET,S
```

This example copies an MS-DOS file (command line parameters 1 through 3) to a source element named BUDGET in the OS/3 library defined with a logical file name of "SRC".

Command line option "X" is specified to cause the input to be "hexified" by the PC software before the data is passed to the TLIB program.

3.15. CPAGE — Set Control Page

The CPAGE program may be used to set the control page "XMIT" setting of a UTS-400 style terminal. The "XMIT" setting of the control page determines the data that is transmitted to the host computer when the terminal operator presses the XMIT key.

Syntax:

```
CPAGE[/opt]
```

Where:

opt Command line option indicating the desired setting of the XMIT option of the control page:

A	Sets the control page to transmit all ("ALL")
V	Sets the control page to transmit variable (unprotected) ("VAR")
C	Sets the control page to transmit changed ("CHAN")

Example:

```
CPAGE,V
```

Additional Considerations:

The preferred option for TIP/30 operation is "VAR". Some IMS programs may require the control page set to "ALL" to permit the IMS program to receive the correct input data from the terminal.

Setting the control page to XMIT=CHAN from the TIP/30 command line has no lasting effect because the TIP/30 Command line Processor (TCP) immediately changes the XMIT setting from CHAN to VAR. This is because XMIT=CHAN is virtually useless except when a screen format is in use.

3.16. CRASH — Abnormal TIP/30 Shutdown

The CRASH program causes TIP/30 to shut down immediately. CRASH does not wait for all users to log off. A JOBDUMP will be taken (assuming the JOBDUMP option was specified in the TIP/30 job control stream). There is normally no need to take a SYSDUMP when TIP/30 terminates abnormally unless instructed to do so by Support Department personnel.

Syntax:

```
CRASH
```

Where:

No parameters required.

Additional Considerations:

The system SHUTDOWN program (if one is specified) is not scheduled. TIP/30 attempts to properly close all files that are open at the time the CRASH program is run.

CRASH is the same as running the STOP transaction (see "3.84. STOP — Shutdown TIP/30 Immediately" on page 3-273) with the exception that a dump is generated.

3.17. CREATE — Create Dynamic File

The CREATE program is used to make a file entry in the TIP/30 catalogue. By doing this, the user is creating a new FCS dynamic file within the TIP$RNDM physical file.

Syntax:

```
CREATE[,type]    aft-name,file-name
```

Where:

type The type of dynamic file to create:

- **P** The file created is a permanent dynamic file and remains in the system after the user logs off (unless it is explicitly scratched).
- **T** The file created is temporary and will be scratched by TIP/30 when the user logs off. In the case of an HPR or other failure, this file will be scratched during the subsequent TIP/30 initialization.

  This is the default type.

aft-name The logical file name to be assigned to the file. After the file has been created, it is automatically assigned to the user.

This is the entry in the active file table (AFT).

file-name The entry to be made in the catalogue for the new file. The catalogue-name consists of three sections, USER-ID/CATL-ID/FILE-ID which uniquely identify each file in the catalogue. The user must at least specify FILE-ID to access the file.

If the USER-ID is not specified, then the user-id of the user using the CREATE program is used.

If CATL-ID is not specified then CATL-ID is set to FILE-ID.

In the following example assume that the USER-ID 'ARC' was used to LOGON.

Example:

```
CREATE    STRTUP,BGNFL
```

Will create the file "ARC/BGNFL/BGNFL" as a temporary dynamic file and assign it the logical name of STRTUP.

3.18. D* — Delete Spooled Data

The D* utility submits an OS/3 console command that will delete ALL spool queues for a specified job.

The submitted command will be logged on the OS/3 console and in the job log for the TIP/30 job.

The command submitted is: **DE SPL,ALL,JOB=xxxxxxx**

Syntax:

```
D*  jobname
```

Where:

jobname The job name that is to be used in the resultant command. This parameter is mandatory!

Additional Considerations:

Refer to the description of the OS/3 operator console command "DE" in the appropriate operating system publication.

3.19. DE — OS/3 DELETE Command

The DE transaction implements the OS/3 "DELETE" console operator command. The DE transaction is actually a clone of the more powerful SYM transaction. The SYM transaction program can be invoked using an alias name that is interpreted as a specific command. When the SYM program is invoked with a transaction name of "DE", the SYM program assumes that the OS/3 command is "DE".

The OS/3 DELETE command syntax is documented in the operation guide for your system.

The DE transaction requires the same syntax as the console command. The command is passed exactly as entered to the OS/3 command processor for execution as if it had been entered at the system console.

Note: There is no provision for returning any completion status.

Example:

```
DE COBCOMP3
```

The above command deletes a queued job named "COBCOMP3".

3.20. DEBUG — Set File in Test Mode

The DEBUG program places a named file in a READ ONLY mode for testing programs. A command line option indicates whether the file is to be placed in debug mode or removed from debug mode. In debug mode, any WRITE attempts from any program (at this terminal) are ignored — the program is given good status, but the physical write operation is not performed!

Syntax:

```
DEBUG,[opt]    lfn
```

Where:

opt The option field is used to indicate whether debug mode is to be set on or set off:

N	Place file in debug mode.
F	Remove file from debug mode (the default).

lfn The logical file name of the file that is to be placed in (or removed from) debug mode.

Example:

```
DEBUG,N  CUSTOMER
```

This command places the file assigned to the logical name of "CUSTOMER" in debug mode and would ignore any subsequent write requests to the file from this terminal.

Error Conditions:

File not assigned.

Additional Considerations:

This option is only effective while the file is assigned to the user (once the file is de-accessed the DEBUG option is no longer effective).

3.21. DEFKEY — Define Function Keys

The DEFKEY utility program allows the user to specify a character sequence that will be "painted" on the screen whenever a function key is pressed as a response to the standard system prompt:

```
TIP?▶
```

After the character sequence is painted on the screen, DEFKEY outputs an auto transmit sequence to the terminal (this automatic XMIT may be suppressed).

The net effect of this is to simulate the keying of that character sequence.

The definition of function key contents is specified by user group. The search for the appropriate function key contents follows the same sequence as the standard order of search in the catalogue: the user's private group is searched first, then elective groups and finally the universal group TIPY.

By utilizing the DEFKEY program, the user may assign character strings to function keys and make it simple to enter the character strings.

The DEFKEY program stores the function key definitions in a TIP/30 dynamic file with the name: group/FUNCTION/KEYS (where "group" is name of the group to which these definitions apply). The function key definitions are retained in this dynamic file until the definitions are explicitly deleted.

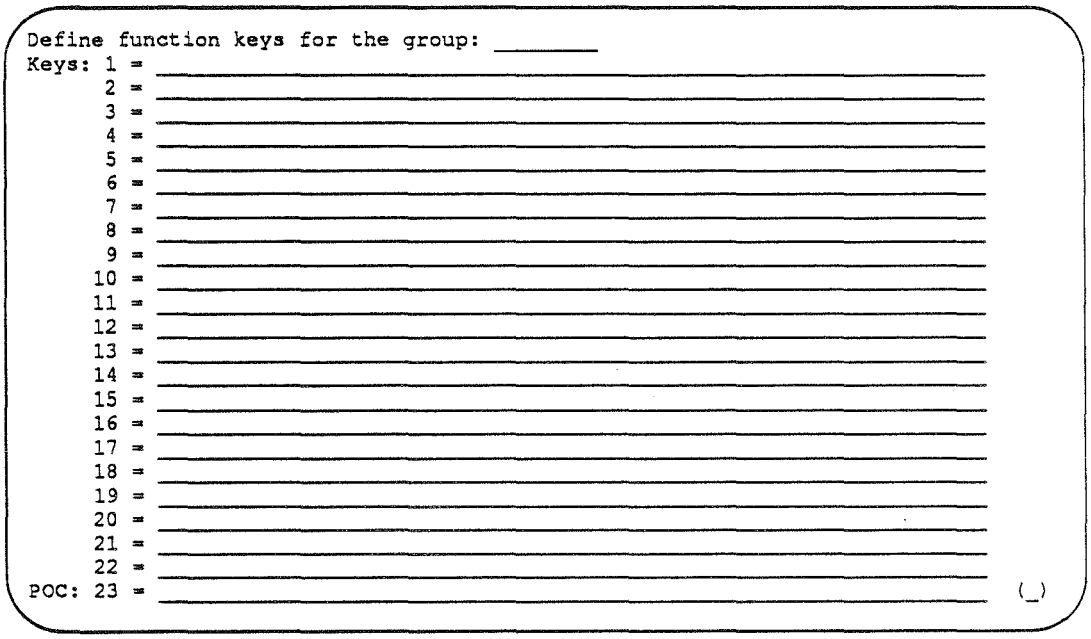
Syntax:

```
DEFKEY  [grp]
```

Where:

grp The name of the group desired. Default is the user's private group.

DEFKEY displays the following screen format:



This screen format is presented (along with the current function key contents — if any) for the specified group name.

The user may change any of the function key contents AND/OR change the group name. If the group name is changed, the DEFKEY program assumes that a NEW function key set is being created from an existing group's definitions.

If the last non-blank character in a function key string is the backslash character [\] the DEFKEY program interprets that as a signal that the automatic XMIT feature is NOT desired when that function key is selected. If that function key is pressed, the character string (without the [\]) is painted on the screen and the terminal operator is then able to make any desired alterations and press the [XMIT] key.

To delete all function key definitions, press the [F4] key. The DEFKEY program displays a message on the screen instructing you to press [XMIT] to confirm that you want to erase the DEFKEY function key file. Pressing any key other than [XMIT] cancels the delete operation.

Example:

```
DEFKEY EDP
```

This command presents the function key definitions currently in effect for the group "EDP" and allows the user to change the definitions.

Additional Considerations:

Function key 23 is a pseudo-function key indication that may be returned by some terminals (for example, the master terminal in a UTS-400 cluster) when a power-on confidence (POC) test is initiated — this is a hardware strapping option of the UTS-400.

SVT terminals may also return such a signal whenever the terminal is reset. The DEFKEY program allows the user to define a transaction to be executed if and when such a signal is received — refer to the description of the POC utility transaction for more information.

WARNING

Transactions may be defined in the TIP/30 catalogue with special 4-character names of the form: F#nn (where nn is a two digit number 01 through 23 inclusive). These transaction names are specially interpreted at the TIP/30 command line to correspond to the associated function key.

Such program catalogue definitions take precedence over any function keys defined using the DEFKEY utility transaction.

3.22. DIE — Abort a Program

The DIE program may by used to force an abnormal termination of a user program running at a terminal.

Syntax:

① `DIE/identifier`

② `DIE identifier`

Where:

identifier The userid or terminal name to be aborted. As indicated by the syntax shown above, this value is required and may be entered either as the command line option field or as command line parameter number 1.

Prefix notation may be used; for example: DIE *BACK$

DIE does not act on the process that is executing; that is, you may not DIE yourself (see "additional considerations" which follow).

Example:

```
DIE JOHN
```

This command causes the program being executed by user JOHN to be abnormally terminated.

Error Conditions:

User or terminal cannot be found.

Additional Considerations:

The program is not aborted immediately if it is swapped out of memory (waiting for terminal input for example). It will be aborted the next time it is running. One may have to press [XMIT] or [MSG-WAIT] on the terminal executing the program to cause TIP/30 to reschedule the program and therefore cause the abort.

In some cases, TIP/30 may not be able to abort the program at all. If this occurs, the operator console unsolicited command "DIE" may prove somewhat more powerful (although that is not necessarily guaranteed).

A program that is running at your terminal can be aborted by entering "@@DIE" as an input message. The letters "DIE" must be in upper case! Whenever TIP/30 detects an input message with the first five characters equal to "@@DIE", the executing program will be aborted with a "Process Cancel Exception" (a bizarre sort of suicide).

This technique can be used to stop a program that is in an input loop with no apparent means to escape from the loop.

Another potential use is to force an abort at a specific input message to allow the programmer to examine the program's work areas.

3.23. DIR — Display Library Directory

The DIR transaction displays a directory of an OS/3 library (or some subset of an OS/3 library) at the terminal. One line is displayed for each module. The line shows the module name, module type, comment and the date and time when the module was last changed.

This transaction is actually a clone of the TLIB transaction. See the description of the DIR command in the TLIB documentation for additional information.

Syntax:

```
DIR  file [,prefix]
```

Where:

file The selected library name as defined in the TIP/30 catalogue.

prefix An element name prefix to be used to select some subset of the elements in the library. If the prefix parameter is omitted, the default is assumed to be all elements.

Example:

```
DIR  JCS,*TJ
```

Displays a directory of all elements with names that begin with "TJ", in the library file catalogued with the name "JCS".

Example of DIR output:

```
Continue?▶Yes  ▶No
 Listing: TIP/*TH$,D  89/02/23  23:00  DIRECTORY
TH$TQLMO,S    VER-002.D HELP FOR 'TQLMON'       84/01/13  14:41
TH$TQLC,S     VER-004.D HELP FOR 'TQL C'        84/01/13  16:11
TH$TQLCO,S    VER-002.D HELP FOR 'TQL CO'       84/01/13  16:16
TH$TQLCP,S    VER-003.D HELP FOR 'TQL CP'       84/01/13  16:18
TH$TQLWP,S    VER-003.D HELP FOR 'TQL WP'       84/01/16  11:29
TH$TQLW,S     VER-003.D HELP FOR 'TQL W'        84/01/16  11:30
TH$TQLUP,S    VER-003.D HELP FOR 'TQL UP'       84/01/16  11:30
TH$TQLUF,S    VER-003.D HELP FOR 'TQL UF'       84/01/16  11:31
TH$TQLUC,S    VER-003.D HELP FOR 'TQL UC'       84/01/16  11:31
TH$TQLU,S     VER-003.D HELP FOR 'TQL U'        84/01/16  11:32
TH$TQLSP,S    VER-003.D HELP FOR 'TQL SP'       84/01/16  11:32
TH$TQLS,S     VER-003.D HELP FOR 'TQL S'        84/01/16  11:33
TH$TQLQP,S    VER-005.D HELP FOR 'TQL QP'       84/01/16  11:34
TH$TQLQ,S     VER-003.D HELP FOR 'TQL Q'        84/01/16  11:34
TH$TQLPP,S    VER-003.D HELP FOR 'TQL PP'       84/01/16  11:35
TH$TQLP,S     VER-003.D HELP FOR 'TQL P'        84/01/16  11:35
TH$TQLNP,S    VER-003.D HELP FOR 'TQL NP'       84/01/16  11:36
TH$TQLNF,S    VER-003.D HELP FOR 'TQL NF'       84/01/16  11:36
TH$TQLN,S     VER-003.D HELP FOR 'TQL N'        84/01/16  11:37
TH$TQLM,S     VER-003.D HELP FOR 'TQL M'        84/01/16  11:38
TH$TQLLP,S    VER-003.D HELP FOR 'TQL LP'       84/01/16  11:39
TH$TQLL,S     VER-003.D HELP FOR 'TQL L'        84/01/16  11:39
```

3.24. DISABLE — Disable Terminals

The DISABLE program is used to logically disable one or more terminals in the network. When a terminal is disabled, the TIP/30 LOGON program does not accept any input from the terminal. If the LOGON program detects input from the terminal, a message is displayed on the terminal indicating that the terminal has been disabled.

The SET transaction program also has this capability, but may not be appropriate for all users because the SET program has other more powerful capabilities.

The DISABLE program displays a message explicitly confirming each terminal that is disabled.

Syntax:

```
DISABLE  [term, ... term]
```

Where:

term — Up to 8 positional parameters. Each parameter represents a terminal name to be disabled.

Each terminal name may be specified by using standard prefix notation (for example: *T3 means all terminals that have a name beginning with the character string "T3").

The use of prefix notation might result in the specification of the terminal that is running the DISABLE program. If this occurs, the DISABLE program does not consider the running terminal as matching the prefix specification.

Example:

```
DISABLE T103,T114,*PD
```

This command disables terminals T103, T114 and all terminals with names that begin with "PD".

Additional Considerations:

The DISABLE program will not allow the user to disable the terminal that he is using to run the disable program. The alternate transaction code ZZDWN is provided for compatibility with IMS.

Error Conditions:

The DISABLE program may report that a terminal name is not valid. This may occur because a terminal name was misspelled or because the terminal name is the terminal that is running the DISABLE program.

3.25. DLL — Down Line Load Utility

The DLL transaction program is designed to assist the user working with the UTS-400 terminal. DLL provides the capability to down line load the UTS-400 memory from the host. The UTS-400 may be loaded with user developed programs or programs produced using Unisys software.

In addition, the UTS-400 may also be loaded with screen formats that have been created with the TIP/30 Message Control System. A supplied UTS-400 program (MCS400) must be loaded whenever the user is down line loading MCS screen formats. Refer to the section on the Message Control System for further information on the use of down line loaded screen formats.

The DLL program may also be called by the transaction code "DLOAD". This transaction code is recognized by the DLL program as a request to down-line load an existing load module to either a UTS-400, UTS-30 or UTS-40.

It is important to note that the operation of this (DLL) program involves a staging buffer within the program. All program and message requests are collected in this staging buffer, then the entire buffer is loaded into the UTS-400 with a single command. The DLL commands are as follows:

Include file/element

Add the UTS-400 object module generated by either ASM80 or UTSASM (described later) to the staging buffer. The transfer address is set to the address specified in the transfer record of this module.

TIP/MCS400 is a module which interfaces with MCS in the host to display screen formats on the terminal.

Get module

Add the UTS-400 object module as generated by either MAC80, PL/M, or UTSCOB to the staging buffer. The transfer address is set to the address specified in the transfer record of this module.

Ntr address

The transfer address is changed to the address specified in the first parameter.

Message mcs-name,[U]

Decode and load the specified MCS message into the staging buffer. Only heading information is normally stored. When this message has been loaded into the terminal, the TIP/30 Message Control System will only send the data portions, thus resulting in a complete screen.

If the second parameter is "U", this message will be stored with fillers to represent the data fields. If you have a data entry screen for which you want the data fields to be filled with underscores, then specify "U" with this command and specify "_" as the filler character in the MCS packet when your program calls TIPMSGO. This will result in the shortest possible XMIT time to display the message on the terminal.

Function key#,word,XMIT,SOE

This equates a UTS-400 function key F5 thru F13 with a character string of up to 8 characters. When the function key is pressed the string will be written on the terminal where the cursor is positioned at that time.

If the 3rd parameter is XMIT, an auto-transmit function will take place.

If the 4th parameter is SOE, a start-of-entry will be placed in front of the character string.

Load [terminal] [,dvc,name]

The contents of the staging buffer will be down line loaded into the memory of the UTS-400. After the UTS-400 has been loaded, the transfer record is sent to the UTS-400. If no terminal is specified, then the down line load is performed on the terminal that is in use. Note that only the master or primary terminal of a UTS-400 cluster receives down line loading.

'dvc' is the auxiliary device index where the program is to be stored (eg: diskette).

'name' is the name to be given to the program when it is stored on the diskette.

LT [terminal] [,dvc,name]

Same command as Load except the time of day from the host will be loaded to location A06B in the format 'HHMMSST', 7 digits of hours, minutes, seconds, and tenths.

R file/elt Redirect input to the given file/element. This command is very useful. The user may make up a canned run stream for this program which may be run at the beginning of each day to load all of the UTS-400 clusters with the screen formats.

This may be on the command line to DLL, for example:

"DLL <file/elt"

or the "R" command may be issued interactively to DLL.

E End execution of DLL.

Note: Only the first letter of the DLL commands (except the "LT" command) is required to identify the command.

Example of DLL command stream:

```
Include TIP/MCS400              -MCS terminal program
Message ACCT1,U                 -accounting screen
Message PAY1                    -payroll screen
Function 5,PAYUPDT,XMIT,SOE     -send in the word PAYUPDT
Loadm   TM01                    -load into UTS-400 cluster
Loadm   TM06                    -load into UTS-400 cluster
End                             -end of loader
```

3.25.1. MCS400 — Message Control System

MCS400 is a UTS-400 program which is written in Intel 8080 assembler language and supplied to the user as an ASM80 object module. It operates under the direction of TIP/30 MCS to display screen formats on the terminal and eliminate the need to continually transmit it down the line.

This program, and messages to be used, should be loaded into the UTS-400 master when TIP/30 is started, and any time the UTS-400 master (or controller) is initialized (ie: via a power-on confidence (POC) test).

Function keys [F5] through [13] can be programmed by DLL. The other function keys perform as follows:

[F14] Beeps the terminal to let you know that the program has been successfully loaded.

[F15] Takes ZZname from home position, looks for "name" in the screen table, then displays it.

[F16] Displays the next screen format in the table. The screen name is displayed in bottom right hand corner.

[F17] Does an erase display and cursor home.

[F18] Re-displays the last screen format used for this terminal. All of these functions operate independently per terminal in the cluster.

[F20] Allows you to set the time of day in the terminal. The program will keep the time of day as HHMMSST in location A06B. To set this time, enter the time at the home position of the terminal and press the [F20] key. Time is kept in hours, minutes, seconds, and tenths of seconds.

[F21] Begins the display of the time of day at the home position of the terminal.

[F22] End the display of time of day.

3.26. DLMSG — Redisplay Last Output

The DLMSG transaction is provided to redisplay the last output message sent to the terminal. This facility is an optional feature of the TIP/30 system that normally is configured only to provide compatibility with the corresponding facility of the Unisys Information Management System (IMS).

The DLMSG program will report "No message was saved" if the program is invoked and the appropriate configuration feature is not specified — see the technical note under "Additional Considerations" below.

Syntax:

```
DLMSG [term]
```

Where:

term Optional command line parameter available only to users with SYSTEM level (or higher) security. If this parameter is supplied and the security constraint is met, the DLMSG will use the specified terminal name rather than the default of the invoking terminal name.

This allows system administrators to examine other terminal's retained output messages.

Additional Considerations:

To use the DLMSG transaction, the TIP/30 system must be configured to include the "TIP$TOM" file.

If TIP/30 detects that a TIP$TOM file is assigned in the job control, TIP/30 retains the last *IMS normal termination output message* for each terminal. The DLMSG transaction may be used to redisplay the last retained output message.

3.27. DLOAD — Down Line Loader

DLOAD is a supplied program designed to assist the user working with the UTS-400 terminal. This program provides the capability of down line loading the UTS-400 memory from the host. The UTS-400 may be loaded with user developed programs or programs produced using Unisys software.

The DLOAD transaction requires one command line parameter specifying the *load module* name of the data that is to be sent to the UTS-400 master terminal.

Syntax:

```
DLOAD loadm
```

Where:

loadm The load module name of the information that is to be sent to the UTS-400 terminal. This load module is normally supplied by the manufacturer, but may be generated by various cross-assemblers available for this purpose.

Trailing zeroes in the load module name need not be entered.

Example:

```
TIP?▶DLOAD U4CPM
```

Additional Considerations:

This transaction is provided primarily for compatibility with IMS systems.

3.28. DOF — Display Open Files

The DOF program displays information about open files for a currently executing OS/3 job.

Syntax:

```
DOF  [jobname] [,file] [,access] [,vsn] [,lbl]
```

Where:

jobname Command line parameter to specify the desired job name (default is the current TIP/30 job).

file Optional filter to select files to be displayed by *filename.*

Prefix notation may be used.

Default is * (all files).

access Optional filter to select files to be displayed by *access.*

Prefix notation may be used.

Default is * (all types of access).

vsn Optional filter to select files to be displayed by *volume serial number.*

Prefix notation may be used.

Default is * (all volumes).

lbl Optional filter to select files to be displayed by *LBL name.*

Prefix notation may be used.

Default is * (all LBL names).

The net effect of all of the filters (parameters 2 through 5) is as if a logical AND operation is applied to all filters.

For example, to display all open files for the current TIP/30 job which have an LFD name beginning with "TIP" and are open with ACCESS=SRD (not beginning with SRD, just SRD), the following command line is required:

```
TIP?▶DOF ,*TIP,SRD
```

Example Output

```
TIP?▶DOF ,*TIP
Lfd name  Access  Volume  File label
========  ======  ======  ==========
TIP$SWAP  EXC     ARCRUN  TIP$SWAP
TIP$CAT   EXCR    ARC103  TIP$CAT
TIP$MCS   EXCR    ARC104  TIP$MCS
TIP$RNDM  EXCR    ARC104  TIP$RNDM
TIP$B4    EXCR    ARCSPL  TIP$B4
TIP$JRN   EXCR    REL100  TIP$JRN
TIP?▶
```

Note: *The word* **WAIT** *may appear in the Access column if the job is presently waiting for a file lock to be released. In this case, the Volume and File Label information for the file that is locked is displayed in the corresponding columns (subject to OS/3 considerations described below).*

In fact, the word WAIT may be used on the command line to display what file a job is waiting for:

```
TIP?▶DOF MYJOB,*WAIT
```

Additional Considerations:

There are inherent race conditions present when running this transaction since it reads information from the system YSHR file.

3.29. DOWN — Set Line Down

This program enables the user to request that a communication line be set down (by ICAM). TIP/30 will request (ICAM) to mark the corresponding line down.

Syntax:

① `DOWN line-name`

② `DOWN term-name`

Where:

line-name The ICAM name of the line that is to be set down.

term-name The ICAM name of a terminal on the line that is to be set down.

Example:

```
DOWN ARC3
```

Additional Considerations:

This program has no effect in a GLOBAL ICAM network (since GUST actually controls the lines and will not honour such requests).

3.30. ENABLE — Enable Terminal Input

The ENABLE program may be used to enable one or more terminals in the network. When a terminal is disabled, the TIP/30 LOGON program does not accept any input from the terminal. The ENABLE program may be used to enable one or more terminals that have been disabled.

The SET program (documented in a separate section) also has this capability, but may not be appropriate for all users because the SET program has other (more powerful) capabilities.

The ENABLE program displays a message explicitly confirming each terminal that is enabled.

Syntax:

```
ENABLE  [terms, ... terms]
```

Where:

terms Up to 8 positional parameters. Each parameter represents a terminal name to be enabled.

Each terminal name may be specified by using standard prefix notation (ie: *T3 means all terminals that have a name beginning with the character string "T3").

The use of prefix notation might result in the specification of the terminal that is running the ENABLE program. If this occurs, the ENABLE program will not consider the running terminal as matching the prefix specification.

Example:

```
ENABLE  T103,T114,*PD
```

This command enables terminals T103, T114 and all terminals with names that begin with "PD".

Additional Considerations:

The alternate transaction code ZZUP is provided for compatibility with IMS.

Error Conditions:

The ENABLE program may report that a terminal name is not valid. This can occur because a terminal name is misspelled or the terminal is the one being used to run the ENABLE program.

3.31. EOJ — TIP/30 Shutdown

The EOJ program starts end of job processing for the TIP system. TIP/30 does not allow any new TIP/30 logons to occur and waits for transactions that are executing to finish.

The EOJ transaction also causes TIP/30 to set the status code PIB-EOJ-PENDING in the Program Information Block (PIB) of all TIP/30 programs that are executing. Well behaved TIP/30 programs periodically check this flag to check whether system shutdown has been requested.

When there are no users remaining on the system, the SHUTDOWN program (if one was specified in the TIP/30 generation or job control) is started.

When all SHUTDOWN programs have finished, all files are closed and TIP/30 terminates normally.

Syntax:

```
EOJ  [ timeout ]  [ WAIT  time-to-eoj ]
```

Where:

timeout The number of minutes to wait for transactions to complete.

If a native TIP program is waiting (eg: via TIPTIMER), the wait time will be adjusted to be the lesser of the actual time remaining and this value.

If an IMS program is in external succession, the maximum time allowed is reduced to this value. This value effectively overrides the TIP/30 generation parameter TIMEOUT=.

WAIT This indicates that the EOJ program is to take effect on a delayed basis (in "time-to-eoj" minutes).

This parameter may only be specified if the EOJ program is run in background (see examples following).

time-to-eoj The number of minutes to defer EOJ processing.

This parameter may only be specified if the EOJ program is run in the background (see examples following).

The EOJ program (running in background!) delays itself (via TIPTIMER) for the specified number of minutes and then proceeds as it normally would.

Example:

```
▶EOJ 5              - Immediate EOJ; set TIMEOUT to 5 minutes.
▶.EOJ WAIT 30       - EOJ in 30 minutes (use TIPGEN TIMEOUT)
▶.EOJ 5 WAIT 30     - EOJ in 30 minutes (set TIMEOUT to 5 minutes)
```

Additional Considerations:

If the EOJ program has been started in background with a delayed effective time (as shown in the last example above), the OS/3 console operator may nullify the pending EOJ by issuing the EOJ console command with the "OFF" option:

```
UNS TIP/30 EOJ OFF.
```

The ability to retract a delayed EOJ command is only available from the OS/3 console (and clearly requires some haste).

3.32. ERASE — ERASE Utility

The ERASE transaction is a clone transaction of the generalized librarian utility transaction TLIB (see separate documentation of TLIB).

The ERASE transaction invokes the TLIB program. When the TLIB program observes that the transaction name is not TLIB, it uses the transaction name as the implied command.

The command line options and parameters that are supplied with the ERASE transaction code are interpreted by TLIB as parameters to the TLIB ERASE command.

The end result is the ability to use ERASE as an apparently stand-alone transaction.

Syntax:

```
ERASE[,options]  parameters
```

Where:

options The command line may contain any command line options recognized by TLIB that pertain to the ERASE command. See description of the options recognized by the TLIB ERASE command.

parameters Parameters required by the ERASE command of the TLIB program.

Example:

```
ERASE  SRC/BUDGET,S
```

This example erases (deletes) a source element named BUDGET in the OS/3 library defined with a logical file name of SRC.

3.33. FCLOSE — Close File(s)

The FCLOSE utility transaction closes one or more online files and marks the files as "unavailable for online use". The files are not available to online programs until a subsequent "FOPEN" is issued (see "3.36. FOPEN — Open Online File(s)" on page 3-98).

This facility is also available as an OS/3 operator console (unsolicited) command to TIP/30 (see section "System Generation and Maintenance").

This program does NOT operate interactively. Up to eight LFD names (or prefixes) may be supplied on the command line; OS/3 Data Management will be presented with a CLOSE request for each file.

Syntax:

```
FCLOSE[/NOW]  lfd1 [,lfd2]  [,lfd3]  ...  [,lfd8]
```

Where:

NOW Command line option to indicate that the specified file(s) are to be forcibly closed if necessary.

If this option is specified, files are closed immediately without any regard for the possibility that the files are in use.

> **WARNING**
>
> Use of this option should be considered a last resort since it may adversely affect transaction roll back and integrity.
>
> Programs that issue TIPFCS calls for a file that has been forcibly closed receive "I/O error" status until such time as the file is reopened.

lfd1..8 The LFD name of the file(s) to be closed; standard prefix notation may be used.

Example:

```
FCLOSE CUSTMAST,INVMAST,ORDENTRY
```

This command closes the three specified files.

Error Conditions:

The LFD name specified may not be a valid LFD name (ie: not in the TIP/30 job control stream or TIP/30 generation).

Additional Considerations:

If the operation is held pending (deferred until all users have relinquished control of the file) the user is not notified when the file is actually closed because the FCLOSE program terminates before the actual Data Management CLOSE function is performed.

Once the FCLOSE is issued, the file is marked as unavailable for online use — new requests to use the file will not be honoured.

The use of this program is logged on the system console and the TIP/30 job log. TIP/30 issues console message number TI076 to alert the system operator that an FCLOSE has been issued.

3.34. FDIR — Display Abbreviated Library Directory

The FDIR transaction displays a "fast" directory of an OS/3 library (or some subset of an OS/3 library) at the terminal. The output display shows the library element name and type code for elements in the library. The display is presented as six columns of information — the display is constructed from left to right and from the top of the screen to the bottom.

The library element comment and timestamp is not included (this information is included in the display produced by the related transaction "DIR").

The FDIR transaction is actually a clone of the TLIB transaction. See the description of the FDIR command for the TLIB utility for additional information.

Syntax:

```
FDIR  file [,prefix]
```

Where:

file The selected library name as defined in the TIP/30 catalogue.

prefix An element name prefix to be used to select some subset of the elements in the library. Default (if omitted) is to list all elements.

Elements are listed without regard to the type of the element.

Example:

```
FDIR  TIP,*TH$
```

Displays a directory of all elements with names that begin with "TH$", in the library file catalogued with the logical file name "TIP".

Example of FDIR output:

```
Continue?▶Yes  ▶No
 Listing: TIP/*TH$,F  89/09/07  09:45  DIRECTORY
TH$TQLMO,S    TH$TQLC,S     TH$TQLCO,S    TH$TQLCP,S    TH$TQLWP,S    TH$TQLW,S
TH$TQLUP,S    TH$TQLUF,S    TH$TQLUC,S    TH$TQLU,S     TH$TQLSP,S    TH$TQLS,S
TH$TQLQP,S    TH$TQLQ,S     TH$TQLPP,S    TH$TQLP,S     TH$TQLNP,S    TH$TQLNF,S
TH$TQLN,S     TH$TQLM,S     TH$TQLLP,S    TH$TQLL,S     TH$TQLDP,S    TH$TQLD,S
TH$QED,S      TH$APTAD,S    TH$APTCU,S    TH$APTLD,S    TH$APTLM,S    TH$APTMT,S
TH$APTSM,S    TH$APTMD,S    TH$DLL,S      TH$APTDE,S    TH$MAS,S      TH$LOGON,S
TH$MSG,S      TH$MAILL,S    TH$WMI,S      TH$SKEL,S     TH$TQL,S      TH$USERS,S
TH$LOOK4,S    TH$SORT,S     TH$SET,S      TH$WARNG,S    TH$PURGE,S    TH$NEWUS,S
TH$GO,S       TH$JBQ,S      TH$ACCES,S    TH$ALLOC,S    TH$APB,S      TH$APTPU,S
TH$BANNE,S    TH$BCP,S      TH$CPAGE,S    TH$SYM,S      TH$CONNE,S    TH$DEBUG,S
TH$DEFKE,S    TH$DOTIN,S    TH$DIE,S      TH$DOC,S      TH$DOF,S      TH$UP,S
TH$LOGOF,S    TH$CCA,S      TH$FREE,S     TH$FSEPM,S    TH$FSE01,S    TH$FSE03,S
TH$FSE04,S    TH$GROUP,S    TH$HARDW,S    TH$HELP,S     TH$HELPE,S    TH$MEM,S
TH$MENU,S     TH$MENUD,S    TH$HANGU,S    TH$NET,S      TH$MSGSH,S    TH$NOTE,S
TH$QCLEA,S    TH$RDR,S      TH$RPG,S      TH$SCR,S      TH$DDPCN,S    TH$SFSCN,S
TH$SHUTD,S    TH$SUBMI,S    TH$SYS,S      TH$RELOA,S    TH$TCB,S      TH$APT,S
TH$FSESU,S    TH$FSEFK,S    TH$FSE02,S    TH$AFT,S      TH$BASIC,S    TH$CALEN,S
TH$CAT,S      TH$CC,S       TH$DBD,S      TH$DCF,S      TH$DD,S       TH$DISAB,S
TH$EDTRS,S    TH$EOJ,S      TH$FCLOS,S    TH$FSE,S      TH$FSE05,S    TH$FSE06,S
TH$ILLTR,S    TH$JCL,S      TH$MENUA,S    TH$MODE,S     TH$MSDOS,S    TH$MSGAR,S
TH$NEWPA,S    TH$PMDA,S     TH$POC,S      TH$SWTCH,S    TH$SYMXX,S    TH$TIPPR,S
TH$VTOC,S     TH$STATU,S    TH$MAIL,S     TH$XFER,S     TH$$$$$$,S    TH$D413,S
```

3.35. FIN — Logoff TIP/30

The FIN program may be used to log off TIP/30. The FIN program is a clone of the LOGOFF program and exists primarily to provide compatibility with older versions of TIP/30.

Syntax:

```
FIN
```

Where:

No parameters are required.

Error Conditions:

If the user has not logged on, TIP/30 will not allow a logoff.

Additional Considerations:

Refer also to the description of the LOGOFF program in "3.53. LOGOFF — Log off TIP/30 System" on page 3-180.

3.36. FOPEN — Open Online File(s)

The FOPEN utility transaction opens one or more files and makes those files available for online program use. The files to be opened may have previously been closed by the FCLOSE transaction (see "3.33. FCLOSE — Close File(s)" on page 3-93).

This facility is also available as an OS/3 operator (unsolicited) command to TIP/30 (OPEN). This program does NOT operate interactively.

Up to eight filenames (or filename prefixes) may be specified on the command line; OS/3 Data Management will be presented with an OPEN request for each file name given.

Syntax:

```
FOPEN[,opt]  file1 [,file2]  [,file3]  ...  [,file8]
```

Where:

opt Optional command line option that may be used for files accessed using CDM (Consolidated Data Management) mode:

INIT	Open file as ",,INIT"
EXTEND	Open file as ",,EXTEND"
SRD	Open file as "ACCESS=SRD"
EXCR	Open file as "ACCESS=EXCR"

The use of SRD or EXCR is restricted to switching from one access to the other (that is, no other access types can be involved — you cannot switch from EXC to SRD for example).

Also see "Additional considerations" which follow.

file1...8 The LFD name of the file(s) to be opened.

Standard prefix notation may be used to open multiple files with a common prefix.

Example:

```
FOPEN CUSTMST,INVMST,ORDENT
```

Opens the three specified files.

Error Conditions:

The LFD name specified may not be a valid LFD name (a spelling error perhaps?).

Additional Considerations:

Note that this program references files by the real LFD name — NOT the catalogued logical file name.

Standard prefix notation applies to the file names; for example, *PAY implies all files with LFD names prefixed by "PAY".

The use of this program is logged on the system console and the TIP/30 job log. TIP/30 issues console message TI076 to alert the system operator that an FOPEN has been issued.

If you intend to make use of the command line options to switch the access that TIP has to files, be sure to NOT include an explicit DD ACCESS= statement in the job control for the file.

The job control "// DD ACCESS=" specification overrides all other specifications and makes it impossible to switch the ACCESS on the run.

3.37. FREE — De-Access a File

The FREE program is used to release a file from assignment to a user. The effect is to remove the file from the active file table for the terminal.

Syntax:

```
FREE[,type] [lfn]
```

Where:

type The type of FREE operation to be performed:

- **A** All assigned files are to be freed. Any temporary dynamic files that are assigned are scratched in this case.
- **F** Any records held for update for the file specified are to be released.
- **X** All records held for update for the user in any file are to be released.

lfn Logical file name — as indicated in the Active File Table (AFT).

Example:

```
FREE  UPDATE
```

Release the file that was assigned with the logical name of UPDATE.

Additional Considerations:

A program that is being tested often may terminate in an untidy fashion: without de-accessing files that were dynamically accessed. When this happens the user can manually FREE the files.

If the TIP/30 command line processor (TCP) regains control, it may display the message "Files still in AFT" to alert the terminal operator that the last program did not properly de-access some of the files that it was using.

3.38. FSE — Full Screen Editor

The Full Screen Editor (FSE) is a screen format oriented editor that is designed to be both powerful and easy to use.

FSE operates by displaying a full screen of text (17 lines) using a TIP/30 screen format. The screen format provides a command area in addition to the display area.

The user may directly alter the text that is displayed or may enter commands to display, find, move, copy, add, delete or modify text in the work space.

Searching and substitution commands are provided and can act on specific column ranges.

Scrolling is accomplished by using the "Forward Page" and the "Backward Page" function keys (or commands).

FSE performs all of its work in a working copy of the data. This copy is held in a TIP/30 Edit Buffer (often called a "work space" in this documentation).

Lines in the work space are displayed with consecutive line numbers that are used as reference points by the various commands.

Syntax:

```
FSE[/R]  [file] [,elt] [,type] [,group] [,buffer] [,reclen]
```

Where:

R Command line option to invoke FSE in "read only" mode.

If this option is specified, FSE inhibits all forms of Write commands.

This option is provided to aid programs that invoke FSE to allow automated use of FSE without risk of accidental library writes.

file The logical file name of the library that contains the element to be read when FSE is invoked.

elt The name of the library element to be read when FSE is invoked.

type The element type code:

D	Detailed library directory (module name, type, comment and time stamp).
F	Fast library directory (module name and type).
I	Internal symbol information (for a load module).
M	Macro.
N	Proc via "name".
P	Proc.
S	Source (default).

group The name of the group that is to contain the edit buffer that is created or accessed.

The default group name is the user's first elective group — as defined in the user catalogue record.

For example: if user MARY belongs to elective groups EDP and PLANNING her default group name is EDP.

buffer The name of the edit buffer to be created or accessed.

Default is "FSE$tttt" where tttt is the name of the terminal where FSE is running.

reclen The length of record that is to be handled by FSE.

The default is 80 for new edit buffers; FSE uses the record length already established for an existing edit buffer.

The record length may range from 80 to 200 (inclusive).

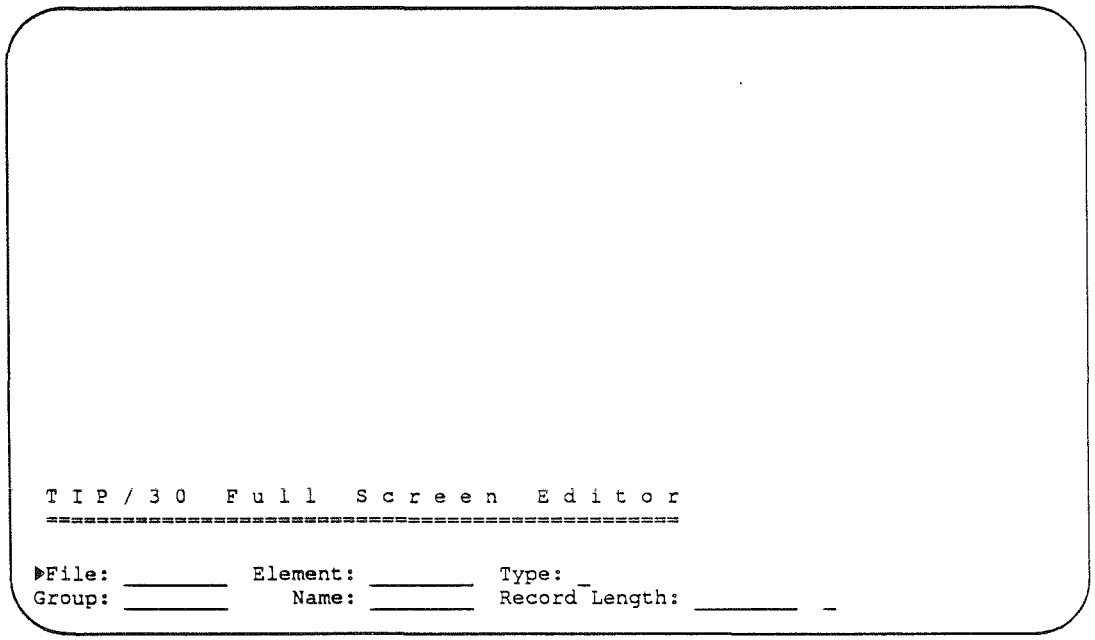
The FSE Write command truncates records at a maximum of 128 characters. OS/3 supports library elements with up to 128 bytes of source text.

Additional Considerations:

If the file, element and type parameters are specified on the initial command line FSE attempts to read the specified element as the initial contents of the edit buffer.

If a single parameter is specified, it is assumed to be the name of a buffer (in the user's first elective group) to be accessed (or created if necessary).

If no command line parameters are supplied, FSE displays the following screen format to allow easy entry of the required information:

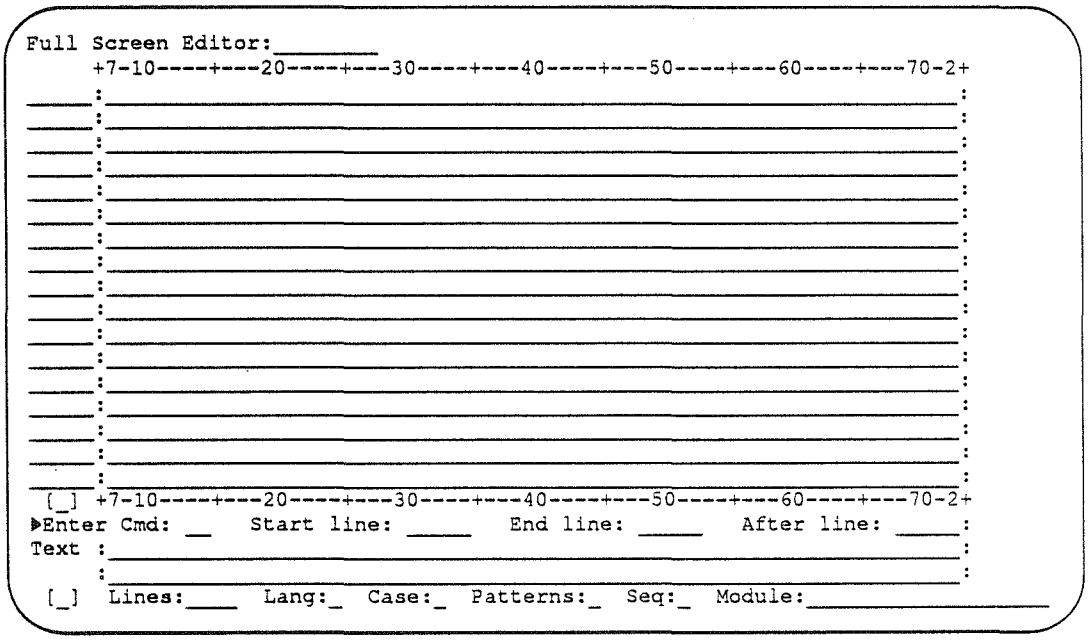


FSE uses a screen format that serves a dual purpose:

1. display the current "page" of text (if any)
2. provide (at the bottom of the CRT) a command and status area.

The screen format displayed varies slightly depending on the type of text in the edit buffer; for simplicity, this documentation assumes that the text has been declared to be COBOL source code.

The screen format appears as follows (underscores are used in this example only to highlight the fields):



The screen format is actually two distinct areas. The upper area (lines 1 through 20) is used to display the current "page" of text from the work space. Line 1 of the screen format contains the current name of the edit buffer and is used for error or informational messages.

The user may also use the upper area for direct modification of the text that is displayed in that area.

The lower area (lines 21 through 24) is used to enter commands to FSE. The last line of the screen format contains protected fields that display current status information about the edit buffer:

Lines This field always indicates the total number of lines in the edit buffer.

Lang This field indicates the declared language type of the data in the edit buffer (see description of the SE command in "3.38.48. SE — Set FSE Options" on page 3-134 for a list of valid language codes).

Case This field indicates the current input case.

"U" — input (alphabetic) translated to upper case.

"L" — input taken literally (no translation).

Patterns A "Y" or "N" indicating whether or not pattern mode is in effect.

In pattern mode, FSE interprets search strings in a non-literal manner.

Seq A "Y" or "N" indicating whether or not FSE is to automatically sequence (according to the declared language) on a write command.

Module The file/elt,type that is currently associated with the work space.

Note: *There are TWO cursor resting positions (the first is near the start of line 20; the second is near the start of line 24).*

The user should be particularly careful about the placement of the cursor before pressing the (XMIT) key.

In this documentation, the command fields are referenced by the following names:

{startline} Field named "Start line:" (row 21)

{endline} Field named "End line:" (row 21)

{afterline} Field named "After line:" (row 21)

{text1} Field named "Text | ... |" (row 22)

{text2} Unnamed field "| ... |" (row 23)

3.38.1. FSE Line Numbers

Line numbers are normally specified to the Full Screen Editor as a positive whole number in the (inclusive) range of 1 to the current maximum line number in the work space. The current maximum line number is always displayed in the lower left corner of the screen.

One exception to this rule is the use of line numbers -9 through -1 (inclusive). A negative line number implies using the line number that was previously "stored" in the line number register 1 through 9 (see description of the FSE "#d" command).

Another exception is the use of line number 9999. That particular value is interpreted as "the last line in the buffer".

Whenever FSE inserts or deletes lines in the work space, the entire work space is renumbered. The work space lines are always whole numbers — fractional line numbers are not used or recognized by FSE.

Note: *FSE recognizes line numbers that are a maximum of four digits (the fields on the screen are defined with a sign — this explains why 5 screen positions exist for each field).*

3.38.2. FSE Column Ranges

A column range is normally specified to the Full Screen Editor in one of two formats:

n A specific column number as a numeric value.

m:n A range of columns from column "m" to "n" inclusive.

The nature of each particular FSE command dictates whether a single column number or a column range is appropriate for the command.

Column numbers in the first format (a single column) are accepted (where appropriate for some commands) in {startline}, {endline}, {afterline}, {text1}, or {text2}.

Column numbers of the second format (a range) are only accepted in the fields {text1} and {text2}.

3.38.3. FSE Strings

When the Full Screen Editor command syntax requires the specification of a text string (for example the "find string" command) the string is normally entered in the {text1} or {text2} field provided in the lower area of the screen format.

The maximum string length that is recognized by FSE is 64 characters (the {text1} and {text2} fields are 72 characters to allow lines that are up to 72 characters to be added).

If the string to be entered includes significant trailing spaces, it must be entered within single or double quotes (either type of quote character is acceptable provided the same one is used at both ends of the string). Leading spaces in a string are always significant.

If the first non-blank character of the string is a digit (0 through 9 inclusive), the string must be entered within quotes (otherwise the digit or digits are assumed to represent a column number or column number range!).

If the string contains a single quote or double quote character, it is suggested that the other quote character be used to delimit the string itself.

When a string is specified as part of a search operation (FInd, DElete, Substitute) the string may be prefixed by an exclamation mark to indicate that a line is desired that does not contain the string (for example: FI !ABC means find the next line not containing the string "ABC").

When the user has set on "pattern matching" mode, some characters in strings take on special meaning — refer to "3.38.65. FSE Pattern Matching" on page 3-146 for the documentation of effects of pattern matching mode.

3.38.4. FSE Command Summary

The following table is a summary of the commands that FSE recognizes. FSE commands are limited to one or two characters. Upper case letters in the command syntax are required; lower case characters are optional.

Table 3-9. FSE Command Summary

Command	Description
Ad	Add lines.
BX	Create comment box.
CB	Copy lines before target line.
CC	Copy column range to another column.
CO	Copy lines.
DE	Delete lines.
DU	Duplicate a set of lines.
En	End editing (retain work space).
EX	Execute TIP/30 command line(s).
FA	Find all lines containing a string.
FI	Find next occurrence of a string.
FM	Same as FI, except place found line in middle of screen.
F#	Equate a command to a user function key (F10-F22).
F-	Same as FI, except search in backward direction.
He	Display HELP information.
In	Insert empty lines into screen (after a line).
IB	Insert empty lines into screen (before a line).
L	List (display) lines on screen.
LL	List (display) last page of work space.
MA	Establish left and right margins.
MB	Move lines before target line.
MC	Move constant or column range to a column range.
MO	Move lines.
O	Set language to "space".

continued ...

Command	Description
OA	Set language "A".
OC	Set language "C".
OD	Set language "D".
OL	Option Literal case (input text unaltered).
OP	Set language "P".
OR	Set language "R".
OT	Set language "T".
OU	Option Upper case (input text mapped to upper case).
OX	Set language to "X".
Pr	Print (display) lines — same as L command.
PE	Peek at a specified line.
P+	Peek at line after last peeked at line.
P-	Peek at line before last peeked at line.
Qu	Quit editing (discard the work space).
Re	Read lines from a library module.
RC	Recall (redisplay) last command entered from keyboard.
Su	Substitute old string with new string.
SA	Sort ascending.
SD	Sort descending.
SE	Set editor defaults.
SP	Substitute old string with new string; show changes.
SW	Switch (interchange) two lines.
Up	Update a range of lines.
Wr	Write lines to a library module.
WE	Write lines to a library module and then END.
WN	Write lines to a library module (NO OVERWRITE PROMPT!).
WQ	Write lines to a library module and then QUIT.
+	Forward n lines.
-	Backward n lines.

continued ...

Command	Description
=	Same as SE command.
<	Shift line data to the left.
>	Shift line data to the right.
^	Call FSE recursively.
#d	Save line number in FSE register number d (1-9).
!d	Clear FSE register number d (1-9).
XMIT	In Command area (with {Cmd} field empty) — display next screen. (Forward Page) In Display area — update lines from display area
MSG WAIT	END editing (save work space if not empty).
F1	Refresh screen display.
F2	Display next screen (Forward Page).
F3	Display previous screen (Backward Page).
F4	QUIT editor; discard work..
F5	Insert blank line ahead of line that cursor is on. — or restore line just deleted with F6
F6	Delete line that cursor is on.
F7	"Split" line at cursor location.
F8	"Join" line to following line at cursor location.
F9	Reissue last FInd command.
F10 ... F22	Available for user definition (via F# command).

3.38.5. Ad — Add Lines

The Add command allows the user to add new lines of text after a specific line number. The user should enter "AD" as the command and may specify a {startline} (an entry in {afterline} is synonymous).

If a {startline} or {afterline} is not specified, FSE assumes that the lines are to be added at the end of the work space.

If the text to be added is two lines or less the user may enter the data directly in the {text1} and {text2} fields of the screen format and press transmit from the second cursor resting location.

If more than two lines are to be added, the user should leave {text1} and {text2} blank and simply specify the line number that immediately precedes the lines to be added (this line number may be entered in {startline} or {afterline}).

FSE re-displays the screen with the contents of the specified line in protected format on the first line of the display, and leaves the remaining 16 lines left unprotected and blank.

The user may then enter any desired text below the (protected) first line and press [XMIT] at the first cursor resting location.

Trailing lines that are entirely blank are not added.

The {endline} is ignored by the ADD command.

If the last line added (line 17) is NOT blank, a fresh screen is displayed to allow entry of more lines.

Pressing [MSG WAIT] while in ADD mode cancels the ADD command.

Pressing [XMIT] without entering any data cancels the ADD command.

Example:

```
▶Enter Cmd: AD   Start line:    10   End line:         After line:      :
Text :     05  FILLER    PIC  X(10).                                      :
     :     05  AMOUNT-C  PIC  S9(7)V99.                                   :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example illustrates adding (only two) lines of text directly after line 10. Since the text to be added was only one or two lines it is more convenient to code them in the text area provided rather than issue a naked AD(10) command and then entering the text in the upper area of the screen format.

3.38.6. BX — Create Comment Box

The BX command may be used to generate a comment box. This command is available for language COBOL or ASSEMBLER.

The {startline} must be set to the line immediately preceding the desired start of the box.

The {afterline} is set to the number of interior lines in the box (3 is the default for this value).

The first character of {text1} may be used to override the character used to draw the box ("-" is the default).

Example:

```
▶Enter Cmd: BX   Start line:    12   End line:         After line:      :
Text :=                                                                   :
     :                                                                    :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

Generates:

```
Full Screen Editor:bufrname
     +7-10----+---20----+---30----+---40----+---50----+---60----+---70-2+
  13 :*=============================================================* :
  14 :*                                                             * :
  15 :*                                                             * :
  16 :*                                                             * :
  17 :*=============================================================* :
     :                                                                :
     :                                                                :
```

After creating the comment box, FSE re-displays the upper screen starting with the first line of the box and positions the cursor in the first blank on line 2 of the box (to facilitate entering the comments).

3.38.7. CB — Copy Lines Before

The CB command allows the user to copy a range of lines from one part of the edit work space to a point that is "before" another line.

Enter "CB" as the command and provide the starting line and ending line to be copied. Specify in {afterline} the line number of the line which is to follow the copied lines.

For example, to copy lines 10 through 80 before line 1, specify the command as "CB", the {startline} as "10", the {endline} as "80" and the before line (in the field called {afterline}) as "1". FSE copies the specified lines before line 1. The lines originally at lines 10 through 80 remain unchanged.

If a string is entered in {text1}, then only lines containing that string are copied. If a column range precedes the string then only lines that contain the string in that column range are copied.

Of course, such a qualification string may be preceded by an exclamation mark ("!") to indicate that the absence of the string is the desired qualification.

For example, to copy lines 10 through 80 before line 1 only if they contain the string "05" in columns 12 through 24, enter "CB" as the command, 10 as the {startline}, 80 as the {endline}, 1 as the {afterline}, and 12:24'05' in {text1} as in the following example:

Example:

```
▶Enter Cmd: CB   Start line:    10   End line:    80   After line:     1:
Text :12:24'05'                                                          :
     :                                                                   :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

3.38.8. CC — Copy Column Range

The CC command allows the user to copy columns of existing text to a target column within a range of lines. The {startline} defines the first line of the range and the {endline} defines the last line of the range.

The {text1} line must contain the column range to be copied. This value must be entered as a column range (ie: 1:5 or 9, etc).

The {text2} line must contain a single column number indicating the target column.

The columns of text specified by the first specification are copied to the column specified (pushing existing text to the right as the copied text is inserted).

The default line range of this command is the 17 lines that are currently displayed in the text area of the screen.

Text is not pushed past the right margin (extra text "falls off" the right edge of a line and disappears into the bit bucket).

Example:

```
▶Enter Cmd: CC   Start line:         End line:         After line:      :
Text :1:10                                                               :
     :40                                                                 :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example illustrates copying columns 1 through 10 to column 40 (effective on the lines that are currently displayed on the screen).

3.38.9. CO — Copy Lines After

The copy command allows the user to copy a range of lines from one part of the edit work space to a point that is "after" another line.

Enter "CO" as the command and provide the starting line and ending line to be copied as well as the number of the line that is just ahead of the desired location of the copied text. The last line in the edit buffer is the default value of {afterline}.

For example, to copy lines 1 through 8 after line 17, the user would specify the command as "CO", the {startline} as "1", the {endline} as "8" and the {afterline} as "17". FSE copies the lines after line 17 and ahead of the line that was line 18. The lines originally at lines 1 through 8 remain unchanged.

If a string is entered in {text1}, only lines containing that string are copied. If a column range precedes the string then only lines that contain the string in that column range are copied.

Of course, such a qualification string may be preceded by an exclamation mark ("!") to indicate that the absence of the string is the desired qualification.

For example, to copy lines 1 through 8 after line 17 only if they contain the string "05" in columns 12 through 24, enter "CO" as the command, 1 as the {startline}, 8 as the {endline}, 17 as the {afterline}, and 12:24'05' in {text1} as in the following example:

Example:

```
▶Enter Cmd: CO   Start line:     1   End line:     8   After line:    17:
Text :12:24'05'                                                          :
     :                                                                   :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

3.38.10. DE — Delete Lines

The delete command allows the user to delete a range of lines. As a precaution, FSE does not allow the user to delete lines UNLESS the first line (of the range) is currently being displayed in the upper area of the display OR has been revealed by a PEEK command.

The user must enter "DE" as the command, the {startline} and an optional {endline}. The Full Screen Editor deletes the lines from the {startline} to the {endline} inclusive.

If an {endline} is not specified, the DE command deletes only one line (the {startline}).

The delete command may be limited to lines that contain a certain string by entering a string in {text1}. Lines may be deleted if they do NOT contain a certain string by prefixing the string with an exclamation mark.

For example, to delete lines 10 through 20 only if they contain the string "VALUE", enter DE as the command, 10 as the {startline}, 20 as the {endline}, and VALUE in {text1}.

Example:

```
▶Enter Cmd: DE   Start line:    10   End line:    20   After line:      :
Text :VALUE                                                              :
     :                                                                   :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example illustrates deleting lines that contain the string VALUE (5 characters) within lines 10 through 20 inclusive.

Lines not containing the string VALUE could be deleted by specifying the string as **!VALUE**

The qualification string may also make use of a column specification to select lines containing (or not containing) the string within a specified range of columns.

3.38.11. DU — Duplicate Lines

The duplicate command allows one or more lines to be duplicated a specified number of times. The duplicated line(s) always immediately follow the original set of lines.

For example, to duplicate lines 5 through 8, 4 times, enter DU as the command, 5 as the {startline}, 8 as the {endline}, and 4 in the {afterline} field:

Example:

```
▶Enter Cmd: DU   Start line:     5   End line:     8   After line:     4:
Text :                                                                   :
     :                                                                   :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

As a safety feature, FSE does not allow the creation of more than 500 new lines with a DU command.

3.38.12. En — End Full Screen Editor

The end command signals FSE that the user has completed all desired editing. FSE terminates normally and retains the work space for potential future editing.

FSE clears the screen and issues a message indicating that the work space is retained. The text of this message includes the group name and buffer name of the retained work space (edit buffer) as a reminder to the user.

(MSG WAIT) is interpreted as the "E" command.

3.38.13. EX — Execute TIP/30 Command Line(s)

The EX command "executes" one or more lines of text as if the text is a TIP/30 command line entry. FSE uses the TIPSUB facility of TIP/30 to execute either a range of lines or one or two lines provided in {text1} and {text2}.

This command eliminates the need to exit the editor (with the "End" command) to be able to run other transactions.

The command line (or lines) to be executed must conform to standard TIP/30 command line structure (no leading spaces; a valid transaction code, etc).

Comment lines (these vary according to the language of the text being edited) are executed — note the transaction name must immediately follow the comment character (eg: "*WMI" starting in column 1 for Assembler or column 7 for COBOL).

Example:

```
▶Enter Cmd: EX   Start line:          End line:          After line:      :
Text :WMI                                                                  :
     :                                                                     :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

-OR-

```
Full Screen Editor:bufrname
     +7-10----+---20----+---30----+---40----+---50----+---60----+---70-2+
   1 :*RV TJ$COB74(FRED),,E=PAY020,F=TSTSRC,USER=FRED                     :
   2 :*.SYS W FRED                                                        :
   3 :                                                                    :
```

```
▶Enter Cmd: EX   Start line:      1   End line:      2   After line:      :
Text :                                                                     :
     :                                                                     :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

3.38.14. FA — Find All Lines Containing a String

The find all command is used to search for all lines that contain a given string. The lines are displayed one page at a time and may be updated by altering them on the screen and pressing [XMIT] with the cursor in the first resting location.

Pressing [F2] displays the next page of matching lines.

Entering another FSE command cancels the "find all" command.

The default starting line number for the "FA" command is the second line that is currently displayed in the upper area of the screen format.

The default ending line number is the last line of the work space.

Example:

```
▶Enter Cmd: FA   Start line:     1   End line:         After line:      :
Text :PIB-                                                               :
     :                                                                   :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example illustrates locating all occurrences of the string 'PIB-". All lines containing the string are displayed (and may be selectively altered and replaced).

Additional Considerations:

The string to be found may be prefixed by a column specification to limit the search for the string to the column range specified. The {text2} field is not used by this command (the search string, with or without a column range, must be in {text1}).

Example:

```
40:60PIC
```

The above command searches for "PIC" starting anywhere in columns 40 through 60 inclusive.

3.38.15. FI — Find Lines Containing a String

The FI command is used to search the work space (in a forward direction — toward the end of the work space) for the next line matching a specified string. A {startline} may be given to start searching from that point.

The FI command does not "wrap around" when it reaches the last line of the work space.

If a line is found that matches the string, FSE re-displays the text — beginning with the line that matched the search string.

The default {startline} for the FI command is the second line currently displayed in the upper area of the screen.

The user may take advantage of this fact by issuing FI repeatedly to "step" through occurrences of a string (since the second and subsequent FI commands begin looking from the line after the first line on the screen, one automatically avoids finding the same line over and over).

To find, for example, the beginning of the PROCEDURE DIVISION in a COBOL program, enter FI as the command, "1" as {startline}, and "PROCEDURE" in {text1}:

Example:

```
▶Enter Cmd: FI   Start line:     1   End line:         After line:      :
Text :PROCEDURE                                                          :
     :                                                                   :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

A more efficient approach is to take advantage of the fact that COBOL division names must begin in column 8:

```
▶Enter Cmd: FI   Start line:     1   End line:         After line:      :
Text :8PROCEDURE                                                         :
     :                                                                   :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

Additional Considerations:

The string to be found may be prefixed by a column specification to limit the search for the string to the column range specified. The {text2} field is not used by this command (the search string, with or without a column range, must be supplied in {text1}).

Example:

```
25LBL
```

Implies searching for the string "LBL" starting in column 25.

3.38.16. FM — Find Lines Containing a String

The FM command functions in the same manner as the FI command, with one difference: when the desired line is found, the matching line is displayed in the middle of the upper area of the screen rather than in line 1 (hence the command mnemonic Find Middle).

If {startline} is not specified, the starting line for the search defaults to the second line that is currently displayed in the upper area of the screen format.

3.38.17. F# — Define Function Key

Some FSE commands may be worth saving. The F# command allows the user to equate some function keys (F10 through F22) to an FSE command. Subsequent use of the defined function key results in the automatic execution of the equated command.

Commands that are equated to function keys are retained until FSE is terminated (via the E or Q command) or until the function key is redefined by the user.

The F# command also allows the option of permanently saving the definitions of keys F#10 through F#22 in your DEFKEY file (see also the description of the DEFKEY transaction earlier in this manual).

For example, assume that you often wish to advance the display by just 8 lines (a sort of half page advance).

The hard way is to enter "+" as the command and "8" in the {startline} field.

To avoid repeatedly keying this command might be to equate (say) function key #13 (a handy lower case function key on a UTS400) to this command.

To accomplish this, enter "F#" as the command and 13 in the {startline} field. FSE responds by displaying the following screen format:

```
            T I P / 3 0   F u l l   S c r e e n   E d i t o r

            U s e r   F u n c t i o n   K e y   D e f i n i t i o n
            ---------------------------------------------------

Fkey: 13  Command: __  Start line: _____  End line: _____  After line: _____

Text: ____________________________________________________________________
      ____________________________________________________________________

                      Msg-Wait - Return to FSE
                            F1 - Refresh screen
                            F2 - Next function key
                            F3 - Prior function key
                            F4 - Save function keys
                          Xmit - Update definition   [_]
```

Notice that the function key that you specified in the main FSE screen (13 in this example) has been carried forward to this screen.

You may now proceed to fill in {startline}, {endline}, {afterline}, {text1} and {text2} as appropriate for the command that you are assigning to the function key you have selected.

Note that the command that is equated to the function key can be quite complex and might include text information (a search string for example).

To continue our example of half-screen forward paging, we enter:

```
            T I P / 3 0   F u l l   S c r e e n   E d i t o r

          U s e r   F u n c t i o n   K e y   D e f i n i t i o n
          ---------------------------------------------------

Fkey: 13  Command: +_  Start line: ____8   End line: _____   After line: _____

Text: ____________________________________________________________________
      ____________________________________________________________________

                    Msg-Wait - Return to FSE
                          F1 - Refresh screen
                          F2 - Next function key
                          F3 - Prior function key
                          F4 - Save function keys
                        Xmit - Update definition   [_]
```

and press [XMIT] to update the definition of F#13 (in this case).

At this point, the definition of F#13 is updated, but the definition of F#13 is purely local to this session. Once you return to the FSE editing screen (more on that in a moment), you find that pressing [F13] results in the automatic execution of the "+" command with "8" in {startline}. The command is executed directly without being displayed, but it may be "recalled" using the RC command.

To define other keys you may use [F2] or [F3] as advertised in the screen format shown above.

To save the function keys in your personal DEFKEY file, press [F4] whenever you have finished defining keys; such saved keys remain permanently in effect for YOU until such time as you discard the DEFKEY file or re-save the FSE function definitions. Whenever you use FSE, those saved function keys are automatically "live" and in effect.

To return to your FSE session (whether or not you have elected to SAVE your function key definitions), simply press [MSG-WAIT].

3.38.18. F- — Find Backward

The F- command functions in the same manner as the FI command, with one difference: the search proceeds from the specified {startline} in a backward direction (that is, toward the beginning of the edit work space).

If {startline} is not specified, the start line for the search defaults to the line that precedes the first line in the upper area of the screen format.

The F- command does NOT "wrap around" when it reaches the first line of the work space.

3.38.19. He — Help for Full Screen Editor

The H command displays help information about FSE commands.

This information is not shown here since it may change from time to time.

The same HELP information may be solicited by using the TIP/30 help processor (see "3.42. HELP — Display Help Information" on page 3-154).

3.38.20. In — Insert Empty Lines (after)

The I command inserts the specified number of blank (empty) lines following the {startline}. These lines can than be filled in and updated by pressing XMIT with the cursor in the first resting location.

The {afterline} field is used to specify the number of lines to insert. If {afterline} is not specified, one blank line is inserted.

For example, to insert 5 blank lines after line 24, enter "I" as the command, 24 as {startline}, and 5 as {afterline}.

Example:

```
▶Enter Cmd: I    Start line:    24   End line:         After line:     5:
Text :                                                                   :
     :                                                                   :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

3.38.21. IB — Insert Empty Lines (before)

The IB command inserts a specified number of (empty) lines ahead of the specified {startline}. These lines can then be filled in and updated by pressing [XMIT] with the cursor in the first resting location.

The {afterline} field is used to specify the number of lines to insert. If {afterline} is not specified, one blank line is inserted.

To insert 3 blank lines ahead of line 20: enter "IB" as the command, 20 as {startline}, and 3 as {afterline}.

Example:

```
▶Enter Cmd: IB   Start line:    20   End line:         After line:     3:
Text :                                                                   :
     :                                                                   :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

3.38.22. Li — List Lines on Screen

The list command displays a range of line numbers in the upper area of the screen. The user must enter the "L" command in the command field and may specify a {startline} or an {endline}.

If both a {startline} value and an {endline} value are not entered, the L command defaults to a display starting with line 1.

If a {startline} is specified, FSE displays lines starting with the specified {startline}.

If an {endline} is specified, FSE displays lines so that the specified {endline} is the bottom line displayed.

Example:

```
▶Enter Cmd: L    Start line:   128   End line:         After line:      :
Text :                                                                   :
     :                                                                   :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

3.38.23. LL — List Last Page

The list last command displays the last "page" of text from the work space. The user must enter the "LL" command in the command field — no other parameters are required or recognized.

This command simplifies the process of moving the display to the end of the current work space.

Example:

```
▶Enter Cmd: LL   Start line:          End line:          After line:       :
Text :                                                                       :
     :                                                                       :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

3.38.24. MA — Margin Set

The margin command establishes new left and right margins for FSE. The new left margin column is specified in {startline} and the new right margin column is specified in {endline}.

If either field is left empty, the default margin (left or right) is used in its place.

The error message: "Margin number exceeds record length" is displayed if the right margin is beyond the declared record length (established when FSE is invoked).

The default margins used by FSE are:

```
Language "COBOL"     : 7 through 72 inclusive
Language "RPG"       : 6 through 74 inclusive
none of the above    : 1 through 72 inclusive
```

The margin command may be used to gain access to columns that are normally not accessible. For example, for COBOL text, one could set the margins to 7 and 80 and then modify data in columns 73-80.

Example:

```
▶Enter Cmd: MA   Start line:      7   End line:     80   After line:       :
Text :                                                                       :
     :                                                                       :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

3.38.25. MB — Move Lines Before

The MB command is identical to the CB (copy before) command with the exception that the moved lines are NOT left in their previous location.

The MB command requires the {startline}, {endline}, and {afterline} numbers be specified.

An optional string may be specified in the first text line to move only lines that match the specified string.

Example:

```
▶Enter Cmd: MB   Start line:    44   End line:    57   After line:   208:
Text :                                                                   :
     :                                                                   :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example illustrates moving lines 44 through 57 (inclusive) before line 208.

3.38.26. MC — Move Constant or Columns

The move columns (move constant) command allows the user to move a range of columns OR a string constant to a range of columns.

The {startline} and {endline} may be specified to limit the scope of the MC command. Default is all lines that are currently displayed in the upper area of the screen.

The {afterline} field is not used by the MC command.

{text1} must contain the character string OR the column range that is to be moved. {text1} may contain:

- a string representing a constant (eg: ABC or '12')
- a single column number (eg: 13)
- a column range (eg: 10:20)

{text2} must contain a single column or a column range representing the column(s) to be altered.

If the text to be moved is not the same length as implied by the receiving column(s), the text is padded with spaces or truncated as appropriate.

Example:

```
▶Enter Cmd: MC   Start line:     1   End line:    10   After line:      :
Text :PICTURE                                                             :
     :40                                                                  :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example illustrates moving the string "PICTURE" (7 characters) to column 40 of line 1 through 10 inclusive. The text that was in columns 40 through the right margin are shifted right 7 positions to accommodate the new text (any overflow falls off the end into the bit bucket).

Example:

```
▶Enter Cmd: MC   Start line:    10   End line:   100   After line:      :
Text :10:20                                                               :
     :40                                                                  :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example illustrates moving the contents of columns 10 through 20 (inclusive) to column 40 in lines 10 through 100 (inclusive).

3.38.27. MO — Move Lines After

The MO command is identical to the copy command "CO" with the exception that the moved lines are NOT left in their previous location.

The move command requires the {startline}, {endline}, and {afterline} numbers be specified.

An optional string may be specified in the first text line to move only lines that match the specified string.

Example:

```
▶Enter Cmd: MO   Start line:    44   End line:    57   After line:   208:
Text :                                                                  :
     :                                                                  :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example illustrates moving lines 44 through 57 (inclusive) after line 208.

3.38.28. O — Set Language " "

The O command sets the language code of the text being edited to space (unspecified). The O command has exactly the same effect as using the SE command (see "3.38.48. SE — Set FSE Options" on page 3-134) and modifying the language field to a space.

The O command does not require (or acknowledge) any entry in the {startline}, {endline}, {afterline}, {text1} or {text2} fields.

3.38.29. OA — Set Language "A"

The OA command sets the language code of the text being edited to "A" (Assembler). The OA command has exactly the same effect as using the SE command (see "3.38.48. SE — Set FSE Options" on page 3-134) and modifying the language field to "A".

The OA command does not require (or acknowledge) any entry in the {startline}, {endline}, {afterline}, {text1} or {text2} fields.

3.38.30. OC — Set Language "C"

The OC command sets the language code of the text being edited to "C" (COBOL). The OC command has exactly the same effect as using the SE command (see "3.38.48. SE — Set FSE Options" on page 3-134) and modifying the language field to "C".

The OC command does not require (or acknowledge) any entry in the {startline}, {endline}, {afterline}, {text1} or {text2} fields.

3.38.31. OD — Set Language "D"

The OD command sets the language code of the text being edited to "D" (Document). The OD command has exactly the same effect as using the SE command (see "3.38.48. SE — Set FSE Options" on page 3-134) and modifying the language field to "D".

The OD command does not require (or acknowledge) any entry in the {startline}, {endline}, {afterline}, {text1} or {text2} fields.

3.38.32. OL — Option Literal

The option literal command provides a fast and simple method to declare that subsequent input text (including any search strings!) is to be interpreted in the case it was entered at the keyboard.

The SE command (see "3.38.48. SE — Set FSE Options" on page 3-134) can also be used to accomplish this change.

The OL command does not require (or acknowledge) any entry in the {startline}, {endline}, {afterline}, {text1} or {text2} fields.

Note: This setting is retained by FSE across edit sessions — this setting is not encoded in the library header for the module.

3.38.33. OP — Set Language "P"

The OP command sets the language code of the text being edited to "P". The OP command has exactly the same effect as using the SE command (see "3.38.48. SE — Set FSE Options" on page 3-134) and modifying the language field to "P".

The OP command does not require (or acknowledge) any entry in the {startline}, {endline}, {afterline}, {text1} or {text2} fields.

3.38.34. OR — Set Language "R"

The OR command sets the language code of the text being edited to "R" (RPG). The OR command has exactly the same effect as using the SE command (see "3.38.48. SE — Set FSE Options" on page 3-134) and modifying the language field to "R".

The OR command does not require (or acknowledge) any entry in the {startline}, {endline}, {afterline}, {text1} or {text2} fields.

3.38.35. OT — Set Language "T"

The OT command sets the language code of the text being edited to "T" (Text). The OT command has exactly the same effect as using the SE command (see "3.38.48. SE — Set FSE Options" on page 3-134) and modifying the language field to "T".

The OT command does not require (or acknowledge) any entry in the {startline}, {endline}, {afterline}, {text1} or {text2} fields.

3.38.36. OU — Option Upper

The option upper command provides a fast and simple method to declare that subsequent input text (including any search strings!) is to be translated into upper case. That is, any alphabetic characters in the text are automatically translated into upper case.

The SE command (see "3.38.48. SE — Set FSE Options" on page 3-134) can also be used to accomplish this change.

The OU command does not require (or acknowledge) any entry in the {startline}, {endline}, {afterline}, {text1} or {text2} fields.

Note: This setting is retained by FSE across edit sessions — this setting is not encoded in the library header for the module.

3.38.37. OX — Set Language "X"

The OX command sets the language code of the text being edited to "X". The OX command has exactly the same effect as using the SE command (see "3.38.48. SE — Set FSE Options" on page 3-134) and modifying the language field to "X".

The OX command does not require (or acknowledge) any entry in the {startline}, {endline}, {afterline}, {text1} or {text2} fields.

3.38.38. Pr — Print Lines

The Print command is identical to the List command described in a previous section. It does not actually print the lines, but displays them on the terminal. This command is provided because some editors interpret a "print" command as a "display on the screen" function.

To print an entire edit work space, one must first "End" FSE (to free the edit buffer!) and then use the TIP/30 utility program "TLIB" to print the edit buffer:

Example:

```
TIP?▶TLIB PR PAY020,,E
```

To print a portion of an edit buffer (work space), the simplest procedure is to use the FSE Write command to write the range of lines to a temporary library element (the library "RUN" is a convenient place to hold such temporary data) and then invoke TLIB (using the FSE EX command!) to print the temporary element.

3.38.39. PE — Peek at Line

The PE command allows the user to view one or two lines without disturbing the lines that are currently displayed in the upper area of the screen format. The command expects a value to be entered in {startline}. This line (and the line that follows it) are displayed in {text1} and {text2}.

This command is useful when you think you remember a line number (how many move or copy commands have been "off by just one line"?).

The (two) lines of text that are revealed in this manner may then be added by entering an Add command (the text is already in the appropriate area for the Add command!).

Example:

```
▶Enter Cmd: PE   Start line:    87   End line:         After line:      :
Text :                                                                    :
     :                                                                    :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example illustrates peeking at line 87 (and 88 — if it exists).

After pressing XMIT (to enter the PE[87] command), the display might look like this (with lines 87 and 88 displayed):

```
▶Enter Cmd: PE   Start line:    87   End line:         After line:      :
Text :     05  FILLER           PICTURE S9(7)V99.                         :
     :     05  FILLER           PICTURE X(4).                             :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

3.38.40. P+ — Peek Scroll Up

The P+ command is a simple method to scroll (in an ascending sense) the lines that have been displayed with a previous PEEK command. For example, if the user issues a P+ command after issuing a PE(13) command, he is shown lines (14,15). A subsequent P+ command displays lines (15,16) and so on.

This command is useful for line fishing (but gets a little tedious if you aren't close the first time).

Example:

```
▶Enter Cmd: P+   Start line:         End line:         After line:      :
Text :                                                                    :
     :                                                                    :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

3.38.41. P- — Peek Scroll Down

The P- command is a simple method to scroll (in a descending sense) the lines that have been displayed with a previous PEEK command. For example, if the user issues a P- command after issuing a PE(13) command, he is shown lines (12,13). A subsequent P- command displays lines (11,12) and so on.

This command is useful for line fishing (but gets a little tedious if you aren't close the first time).

Example:

```
▶Enter Cmd: P-   Start line:          End line:          After line:       :
Text  :                                                                      :
      :                                                                      :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

3.38.42. Qu — Quit FSE

The quit command causes the Full Screen Editor to end the editing session AND discard the edit work space.

If changes have been made to the contents of the work space and the changes have not been written, FSE warns the user and asks for confirmation of the QUIT command.

The QUIT command is normally used after a write command has copied the work space to a library (to save the contents of the work space).

Pressing F4 is equivalent to this command.

3.38.43. Re — Read from Library or Edit Buffer

The read command allows lines to be read from a library element or another edit buffer.

A {startline} and/or {endline} may be specified to indicate that only a subset of the element (or edit buffer) is to be read.

The {startline} defaults to 1 if it is not specified; {endline} defaults to 9999 (meaning the last line of the input).

The lines that are read are copied immediately after the line number specified in the {afterline} field of the command (or at the end of the current work space if no {afterline} is specified).

The library name, element name and element type may be entered in {text1} in the usual format: library/element,type.

To specify that the Read command is to read from an edit buffer, the parameters are: name,,E (pseudo-type "E" indicates Edit Buffer).

Alternatively, the SE command (see "3.38.48. SE — Set FSE Options" on page 3-134) may be used before using the Read command to specify the library name, element name, and type.

If these fields are provided in the {text1} line of the screen format, any omitted fields are defaulted from the "current" information (in the lower right area of the screen format).

Example:

```
▶Enter Cmd: R    Start line:          End line:          After line:   100:
Text :TSTSRC/PAY020                                                      :
     :                                                                   :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:TSTSRC/PAY035,S
```

This example illustrates reading (all lines by default) of the source element "PAY020" from library "TSTSRC". The lines read in are placed immediately following line 100 of the work space.

Additional Considerations:

A read command is considered to be a "change" of the contents of the work space and therefore causes a subsequent Quit command to think that changes have been made.

3.38.44. RC — Recall Last Command

The RC command re-displays the last command that was entered in the command area (lower portion) of the FSE screen format.

Some FSE commands clear the lower area of the screen format; the RC command allows the user to recall the last command (and therefore resubmit the command with or without modification).

3.38.45. Su — Substitute Text

The substitute command allows the user to replace one string with a different string. The starting and ending lines may be specified to limit the substitution to just that range of lines.

The number of lines modified is reported when the substitute command is completed.

If an optional occurrence number is NOT specified (in the {afterline} field), the substitute command changes all occurrences that are encountered within the range specified.

If neither a {startline} nor {endline} is given then substitution takes place only within the 17 lines currently displayed in the upper area of the screen.

The {afterline} field may contain the desired "occurrence" number (all occurrences on each inspected line is the default).

To substitute the word 'RED' with 'GREEN' in lines 30 through 80, enter "S" as the command, 30 as {startline}, 80 as {endline}, "RED" in {text1}, and "GREEN" in {text2}.

The substitute command processes an individual occurrence of the specified string by specifying an occurrence number in the {afterline} field. In the above example, if only the third occurrence of RED is to be changed to GREEN, enter 3 as the {afterline}.

The substitute command may be further restricted to look only in certain columns for the string. This is done by preceding the old string (in {text1} by a column range.

To change RED into GREEN if the string RED started in columns 15 through 21, enter **15:21RED** in {text1}.

Example:

```
▶Enter Cmd: SU   Start line:     1   End line:   100   After line:      :
Text :15:21RED                                                           :
     :GREEN                                                              :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example illustrates performing a substitution on lines 1 through 100 (inclusive). If the string "RED" is found in columns 15 through 21 (inclusive) of a line it is replaced with "GREEN".

3.38.46. SA — Sort Ascending

The sort ascending command sorts lines into ascending order using a specified column as the start of the "sort key". The sort uses the standard EBCDIC character set as the collating sequence.

The sort is NOT a "stable" sort — lines that have identical "sort keys" do not necessarily remain in their original sequence (with respect to each other).

The user must specify a {startline} and {endline} and may specify in the {afterline} field the column that is to be considered the first column of the "sort key".

If {afterline} is omitted, the sort uses the entire line as the sort key.

A sort command is considered to be a "change" of the contents of the work space (whether or not any lines actually are moved!) and therefore causes a subsequent Quit command to think that changes had been made.

FSE invokes the transaction code "SORT" (see "3.81. SORT — Sort Edit Buffer" on page 3-249) to accomplish the line sorting — the SA command fails if the user does not have proper security to run the SORT transaction.

Example:

```
▶Enter Cmd: SA   Start line:     1   End line:    40   After line:    35:
Text :                                                                  :
     :                                                                  :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example sorts lines 1 through 40 inclusive into ascending sequence, using a key that starts in column 35.

3.38.47. SD — Sort Descending

The sort descending command sorts lines into descending order using a specified column as the start of the "sort key". The sort uses the standard EBCDIC character set as the collating sequence.

This command is identical to the SA command (described in the previous section) except that the lines are ordered in descending sequence by the specified key.

3.38.48. SE — Set FSE Options

The set command allows the user to specify information about the edit work space or to alter certain session parameters for FSE.

The following screen format is displayed when the SE command is entered. The user may change any of the values in the unprotected fields; protected fields (suffixed by an "*" in the discussion that follows) are shown for information only.

```
              T I P / 3 0   F u l l   S c r e e n   E d i t o r
              ================================================

Module;        File: ________ Element: ________     Type: _     Changed: _
             Comments: ____________________ ________ _____

              Version: ___      Language: _          Case:U_

  Pattern Matching: _   Auto Sequence: _   Update Stamp: ________

Buffer;        Group: ________    Name: ________

    Default string: ____________________________________________________

         Page Size: __   Left Margin: ___  Right Margin: ___

    Line registers: ______  ______  ______  ______
                    ______  ______  ______  ______  ______

    Line register display option: _

   Press MSG-WAIT to return to FSE  or  Press XMIT to alter information   (_)
```

File The library name associated with the contents of the edit work space.

This library is the library used by default by the FSE read and write commands.

Element The element name associated with the contents of the edit work space.

The FSE read and write commands assume that this is the default element name.

Type The type of the library element.

FSE read and write commands use this entry as the type of the library element to be read or written.

Comments This field contains the current comments that are maintained in the library directory entry for the element.

Changed This field contains a "Y" if the edit buffer has been changed and those changes have not yet been written to a library element.

Version The current version number of the element.

Language The language code for the contents of the edit work space.

Specifying the language of the text in the work space establishes default margins and dictates the number of columns that FSE displays on the

terminal.

"A" — Assembler format (margins 1-72), text automatically tabbed so that the label field starts in column 1, the opcode field starts in 10, the operand field starts in column 20 and comments start in column 39.

"C" — Cobol format (margins 7-72)

"D" — Documentation (margins 1-72)

"P" — Cobol format (margins 7-72) except that comment lines (* in column 7) are NOT translated to upper case.

"R" — RPG format (margins 6-72)

"T" — Assembler format (margins 1-72), text automatically tabbed so that the label field starts in column 1, the opcode field starts in 10, the operand field starts in column 16 and comments start in column 40.

"X" — Upper case text (margins 1-72)

" " — Upper case text (margins 1-72)

If the language type is changed during an FSE session by altering the information shown by the SE command, the margins are not reset at the same time.

Case The case processing of input text (default is Upper case unless Type="D")

"U" — input forced to upper case

"L" — input may be either upper or lower case

Pattern Matching

"Y" or "N" indicating whether or not pattern matching mode is in effect.

When pattern matching mode is in effect, certain characters in strings take on special meaning. See description of pattern matching in "3.38.65. FSE Pattern Matching" on page 3-146.

Auto Sequence

"Y" or "N" indicating whether or not FSE is to automatically sequence each line (according to its language code) on a write to a library.

Update Stamp

The desired type of "stamping" of updated lines. FSE marks each line that is updated with a specific type of "update stamp".

Update stamping applies only to edit buffers that are marked as language code A, C, P or D.

All other language codes cannot make use of update stamping.

Default depends on TIP/30 generation parameter EDiTstmp=.

"STANDARD" — implies that FSE places the current version number (3 digits) in columns 73-75 of any line that is updated and places an asterisk in column 76 of updated lines (the asterisk reveals lines that were updated during the last editing session).

spaces — same as STANDARD.

"NO" — do not stamp updated lines in any manner

"DATE" — place current date in YYMMDD format in columns 73:78 of updated lines

"USERID" — place current user-id in columns 73:80 of updated lines

none of the above — FSE takes the 8 characters in this field (a secret code??) and uses those characters to stamp columns 73-80.

Important note! The "stamping" process takes place as a side effect of a write command (therefore, updates that occur across more than one edit session receive the same version number).

Since the "stamping" occurs as a side effect of a write command, it follows that the stamping cannot be previewed by looking at columns 73-80 of the edit buffer.

Group* Name of the user group that "owns" this edit buffer (work space).

This normally is the user's first elective group.

Name* The name of the edit work space.

Default String*

The last string used in a search expression.

This is also the default string used if one is not provided on a Find command (FA, FI, FM, F9).

Note that a column specification that was part of a search string (eg: 1:20'FOO') is also retained by FSE.

Page Size The number of lines that constitutes a "page" of text.

FSE moves ahead or back this number of lines when the user uses the Page Forward/Backward commands.

Left margin*

The current column number that is the left margin.

Right margin*

The current column number that is the right margin.

Line registers*

Line numbers that have been "saved" in FSE registers 1 through 9 (refer to #d command).

Display option

Line register display option — whether or not the lines that have been "saved" in a line register (via a #d command) are to be highlighted.

If this field contains a "Y", FSE displays the register number of a saved line number rather than the actual line number (see examples in the discussion of the #d command).

The register number is also embellished with a trailing negative sign (to catch your eye) and the line number field is set to reverse video (although many terminals cannot respond to that attribute!).

Default: "Y" for all languages except RPG.

The RPG screen does not have room for a trailing minus decoration; reverse video is the best one can expect when the language is set to "R".

3.38.49. SP — Substitute and Display Changes

The SP command has the same parameters and options as the regular substitute command. The SP command however, displays the changes that it makes (in the upper area of the screen).

When the screen gets full (after each 17 lines of changes):

- pressing key [F2] simply continues the SP command;
- entering another command terminates the SP command and proceeds to the new command;
- the user may make additional changes (manually) in the upper area of the screen format and press the [XMT] key to make those additional changes — and then press the [F2] key to continue the original SP command.

Additional Considerations:

Since the SP command operates interactively, it does not report the number of lines that were modified as a result of substitution.

3.38.50. SW — Switch (exchange) Two Lines

The SW command exchanges two lines. The line numbers of the two lines to exchange are supplied as the {startline} and {endline}. The {startline} must be less than {endline}.

Example:

```
▶Enter Cmd: SW   Start line:   100   End line:   101   After line:      :
Text :                                                                   :
     :                                                                   :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

3.38.51. Up — Update Line Range

The update command displays a range of lines in the upper area of the screen format and then allows the user to update the lines in place and (possibly) add additional lines.

If a line range is not specified (a naked U command) only line one is displayed for update.

If a line range is specified that represents more than 17 lines (the maximum number of lines that can be displayed in the upper area) FSE displays only the first 17 lines of the range.

Example:

```
▶Enter Cmd: UP   Start line:    70   End line:    74   After line:      :
Text :                                                                    :
     :                                                                    :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This command would result in a display similar to the following:

```
Full Screen Editor:BUFFER
     +7-10----+---20----+---30----+---40----+---50----+---60----+---70-2+
  70 :--text of line 70--                                                 :
  71 :--text of line 71--                                                 :
  72 :--text of line 72--                                                 :
  73 :--text of line 73--                                                 :
  74 :--text of line 74--                                                 :
     :                                                                    :
```

The user may now alter the text in the upper area of the screen (including typing additional lines following the ones shown) and press XMIT to replace the original lines 70-74 with the new text from the upper area of the screen.

3.38.52. Wr — Write Module to Library

The write command allows text to be written out to a library element from the work space. The library name, element name and type may be entered via a previous "SEt" command or in the {text1} field in the lower area of the screen format.

A range of lines may be specified to be written; the default is to write the entire contents of the work space to the indicated library element.

If the library name, element name, or type is omitted in the {text1} field FSE uses whatever values are currently in effect (see "3.38.48. SE — Set FSE Options" on page 3-134) — as echoed in the lower right corner of the screen format. If directory information is currently in the work space (type code "D" or "F"), an omitted type specification defaults to "S", since writing directories is not permitted.

To update the module comment (that is written in the library header for the element), one can either use the SE command before issuing the Write command or specify the comment directly in the first 20 characters of the {text2} field.

If the specified element (of the corresponding type) already exists in the library, FSE displays an informational message indicating that the element exists and requires the user to press the F2 key to confirm that the element is to be overwritten.

If the user DOES NOT press F2 (or spends more than 60 seconds thinking about it), the Write command is cancelled.

Example:

```
▶Enter Cmd: WR   Start line:          End line:          After line:       :
Text :TSTSRC/PAY020                                                          :
     :a comment                                                              :
 [_]  Lines: 108  Lang:C  Case:U  Patterns:N  Seq:Y  Module:PRODSRC/PAY,S
```

This example illustrates writing (the entire work space) to the library "TSTSRC" as an element named "PAY020" (note the somewhat frivolous comment provided in {text2}.

Additional Considerations:

If FSE displays the error message "Error W processing write command", it is possible that the target library is defined in the TIP/30 catalogue with the specification WRITE=NO (meaning that write operations are not permitted).

3.38.53. WE — Write Module to Library and End

The "write with automatic end" command performs the same function as the W command and then performs an END command after the write has completed. This is equivalent to issuing a W command and then separately issuing an E command.

If the specified element (of the corresponding type) already exists in the library, FSE displays an informational message indicating that the element exists and requires the user to press the F2 key to confirm that the element is to be overwritten.

If F2 is NOT pressed (or a 60-second timeout occurs), the WE command is cancelled.

3.38.54. WN — Write (No Overwrite Prompt)

The "write with no overwrite prompt" command performs the same function the W command (see previous section) with the exception that NO prompt is issued if the specified library element already exists.

3.38.55. WQ — Write Element to Library and Quit

The "write with automatic quit" command performs the same function as the W command (see earlier section) and then performs a QUit command after the write has been finished. This is equivalent to issuing a W command and then separately issuing a QUit command.

If the specified element (of the corresponding type) already exists in the library, FSE displays an informational message indicating that the element exists and requires the user to press the F2 key to confirm that the element is to be overwritten.

If F2 is NOT pressed (or a 60-second timeout occurs), the WQ command is cancelled.

3.38.56. + — Forward Space Lines

The forward space command allows the user to go forward by a specified number of lines. Enter + as the command and enter a number in the {startline} field indicating the number of lines to move forward.

Default is one line.

Example:

```
▶Enter Cmd: +    Start line:    10   End line:         After line:      :
Text :                                                                    :
     :                                                                    :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example illustrates moving the display "forward" ten lines.

3.38.57. - — Backward Space Lines

The backward space command allows the user to go backward by a specified number of lines. Enter - as the command and enter a number in the {startline} field, indicating the number of lines to move backward.

Default is one line.

Example:

```
▶Enter Cmd: -    Start line:    10   End line:         After line:      :
Text :                                                                    :
     :                                                                    :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example illustrates moving the display "back" ten lines.

3.38.58. = — Set Options

The = command is identical to the "SE" command previously described and is provided primarily for typing convenience.

3.38.59. < — Shift Display Left

The SHIFT LEFT command ("<") alters the visible portion of the record that is displayed in the upper area of the screen format. The user must specify the number of columns to shift the display in the {startline} field of the command area.

The display is shifted to the left by the number of columns specified OR until the right most column of the record is shown on the screen.

Example:

```
▶Enter Cmd: <    Start line: 8       End line:         After line:      :
Text :                                                                    :
     :                                                                    :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example illustrates shifting the display to the left by 8 columns. If the work space was declared to be COBOL language, the display would now show columns 15-80 instead of the (usual) 7-72.

3.38.60. > — Shift Display Right

The SHIFT RIGHT command (">") alters the visible portion of the record that is displayed in the upper area of the screen format. The user must specify the number of columns to shift the display in the {startline} field of the command area.

The display is shifted to the right by the number of columns specified OR until the left most column of the record is shown on the screen.

Example:

```
▶Enter Cmd: >    Start line: 8       End line:         After line:      :
Text :                                                                    :
     :                                                                    :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example illustrates shifting the display to the right by 8 columns.

3.38.61. ^ — Call FSE Recursively

The CALL FSE command ("^") allows the user to invoke another copy or instance of FSE to start another editing session without having to close the current editing session.

The current editing session (or instance of FSE) is saved by the system and is reactivated when the higher level session is terminated.

This command is especially useful to permit the user to temporarily preempt the current editing session and invoke FSE to edit something else and then return.

If desired, the FSE command line parameters to read a library element may be placed in the first text line area (see example that follows). If no parameters are supplied, FSE is invoked without parameters and FSE reacts by displaying the initial entry screen.

Example:

```
▶Enter Cmd: ^   Start line:          End line:          After line:        :
Text :TIP/TT-TSP,S                                                          :
     :                                                                      :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

This example illustrates invoking another instance of FSE to edit the element TT-TSP from the library TIP.

3.38.62. #d — Saving Line Numbers

FSE maintains nine "registers" (1-9) that may be assigned line numbers. These registers can then be used in place of absolute line numbers in other commands.

The "#d" command is used to save a line number in register "d".

For example, line number 107 can be stored in register number 5 by issuing the "#5" command with "107" in the {startline} field of the command area.

Thereafter, the user can reference that line by specifying line number "-5". For example LI(-5) is interpreted as "list lines beginning with the line number stored in register 5".

If {startline} is omitted, the line number that is currently the first line in the upper portion of the display is assumed.

FSE updates the actual line number in the register to "track" the line. If the location of the line changes due to the occurrence of a line delete or line add in front of the line, the register is updated to reflect the line's new position.

If a line has been noted in a register and the line itself is deleted, FSE clears the register reference for that line.

Example:

```
▶Enter Cmd: #5   Start line:   107   End line:         After line:      :
Text :                                                                    :
     :                                                                    :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

Depending on an option that is present on the option page, the user may or may not observe that saved line numbers are highlighted in the upper display.

When the line register display option is on (set to "Y"), FSE displays the register number instead of the actual line number. The display has the "reverse video" attribute on (for terminals with that capability) and includes a trailing minus sign (for terminals that do not support reverse video).

Example:

```
Full Screen Editor:bufrname
     +7-10----+---20----+---30----+---40----+---50----+---60----+---70-2+
 343 :          MOVE  SPACE  TO  CUSTOMER-MASTER.                        :
   5-:          MOVE  ZERO   TO  CUSTOMER-NUMBER.                        :
 345 :                                                                    :
```

The above example shows line number 344 when it has been noted in line register #5 (many terminals cannot display fields in reverse video — the trailing minus sign is more obvious).

3.38.63. !d — Clear FSE Registers

FSE maintains nine "registers" (1-9) that may be assigned line numbers (see previous discussion of the #d command).

To clear one (or all) of the nine registers, the "!" command is provided.

An "!" followed by a line register number from 1 through 9 clears the specified register number.

Specifying register zero (0) after the "!" clears all line registers.

Example:

```
▶Enter Cmd: !1   Start line:         End line:         After line:      :
Text :                                                                    :
     :                                                                    :
 [_]  Lines: 904  Lang:C  Case:U  Patterns:N  Seq:Y  Module:
```

3.38.64. FSE Function Key Usage

The Full Screen Editor recognizes certain function keys as special commands. Invalid function keys result in an error message on the screen.

[XMIT] The interpretation of the [XMIT] key depends on the location of the cursor:

If the cursor is in the upper area of the display, FSE reads the text in the upper area of the display and alters the corresponding text in the work space (if the text was altered).

If the cursor is in the lower area of the display FSE attempts to process the command. If there is no command, FSE assumes the default command ([F2] — display next screen full).

[MSG WAIT] Pressing [MSG WAIT] is equivalent to entering an "End" command.

FSE terminates and the work space is retained.

[F1] Function key 1 causes FSE to resend the last screen that was output. This may be important if the screen display was altered unintentionally or by the receipt of an unsolicited message.

[F2] Function key 2 is the "Forward Page" key. When this function key is pressed FSE displays the next "page" of source lines.

The number of lines that a "page" implies is a user-definable number of lines (see the description of page size in "3.38.48. SE — Set FSE Options" on page 3-134).

[F3] Function key 3 is the "Backward Page" key. When this function key is pressed FSE displays the previous "page" of source lines.

The number of lines that a "page" implies is a user-definable number of lines (see the description of page size in "3.38.48. SE — Set FSE Options" on page 3-134).

[F4] Function key 4 signals the FSE program that the user wishes to abort the edit session (equivalent to the QUIT command).

If changes have been made since the last time the contents of the edit work space was written, FSE displays a warning message and requires the user to press the [F2] key to confirm that the QUIT is to be performed.

[F5] Function key 5 causes FSE to insert one blank line ahead of the line where the cursor is resting.

The cursor must be placed on the appropriate line in the upper area of the screen before pressing [F5].

The cursor remains at the location where the user pressed [F5].

If [F5] is pressed immediately following the use of [F6] (see next discussion), FSE inserts the line that was deleted by [F6] (a bizarre but deliberate reinstatement of a deleted line).

Note: *This command may not function properly on terminals that do not correctly emulate a UTS400 (so-called plug compatibles).*

F6 Function key 6 causes FSE to delete the line where the cursor is resting.

The cursor must be placed on the appropriate line in the upper area of the screen before pressing F6.

The cursor remains at the location where the user pressed F6.

If F6 is pressed unintentionally, simply press F5 to reinstate the erroneously deleted line (you have one chance to do this — don't waste it!).

Note: *This command may not function properly on terminals that do not correctly emulate a UTS400 (so-called plug compatibles).*

F7 Function key 7 causes FSE to "split" the line where the cursor is resting at the point where the cursor is located.

The original line is turned into two lines.

The first line contains the characters up to (**but not including**) the character under the cursor.

The second of the two lines contains the remaining characters of the first line.

After the line is "split" the cursor is restored to the original split point.

Note: *This command may not function properly on terminals that do not correctly emulate a UTS400 (so-called plug compatibles).*

F8 Function key 8 causes FSE to "join" the line containing the cursor and the line following at the point where the cursor is positioned.

This function is the inverse of F7.

After the line is "joined" the cursor is restored to the original cursor position.

Note: *This command may not function properly on terminals that do not correctly emulate a UTS400 (so-called plug compatibles).*

F9 Function key 9 is considered as a request to reissue the last find command (FI, FM, F-).

This means that another search begins for the string specified in the last search command (if the previous string included a column specification, that too is remembered).

The direction of the previous search command is remembered and honoured provided no other command has been entered at the keyboard (the default direction is reset to "forward" when any command other than a "find" is issued).

F10 — F22

Function keys 10 through 22 are user-definable. The F# command (described in a previous section) may be used to assign a specific command (with or without parameters) to a function key in this range.

That function key may then be pressed to invoke the command that was "soft coded" as that key.

3.38.65. FSE Pattern Matching

When specifying a search string, the desired string is often not explicitly known. Instead, the user is aware of the layout or structure of the string that is desired.

For example, you may wish to search for a particular type of field name that you know is constructed as follows: an alpha character followed by two digits followed by two more alpha characters (such as "A12DE").

To search for that type of string explicitly is not possible — there exists a large number of permutations of characters and digits that satisfy the stated rule.

To address this issue, FSE allows the user to enter "pattern matching mode". In this mode, it is possible to specify search strings in an implicit manner by specifying a desired pattern of characters.

When pattern matching mode is enabled, several characters have reserved meaning and are not interpreted literally by FSE. These characters are referred to as "meta characters".

. (period) The period character may be used to "match" any character (essentially a wild card character).

% (percent) The percent character may be used to "match" any alpha character ("a" through "z" or "A" through "A" through "Z").

 # (pound) The pound sign character may be used to "match" any digit (0 through 9).

*** (star)** The star character (asterisk) indicates that the match pattern is to allow 0 or **more repetitions** of the character which preceded the star

For example: #* would match 0 or more digits in a row.

" (quote) The double quote character indicates that the character which follows is to be taken literally and not interpreted as a reserved character in this instance.

This mechanism allows you to use the period, percent, pound, star (asterisk), double quote, left square bracket or right square bracket characters literally in a search string.

For example, to look for the string "A#B" when pattern matching is turned on, specify A"#B as the search string.

[ ] Square brackets are used to enclose "tag expressions". More on that subject follows.

With these reserved characters, some reasonably sophisticated search patterns can be specified:

Example:

%##%% This is the example discussed at the start of this section. An alphabetic character, followed by 2 digits, followed by two alphabetic characters.

MATCHES: A12DB Z00XX R12BW

(.*) Any number of arbitrary characters (including none at all!) enclosed in parentheses.

Note that this pattern literally implies: a (character, followed by zero or more occurrences of any character, followed by a) character.

MATCHES: (12), (more or less), (text1), (xxxxxxx) and ()

##/##/## Date format.

MATCHES: 86/01/05 and 05/01/87

##"# Two digits followed by a # character.

MATCHES: 37# and 12#

#*"# Any number of digits (including no digits) followed by a # character.

MATCHES: 37#, 12#, 1234567#, 000034# and #

An advanced feature of pattern matching mode is the ability to segregate a portion of a search string and assign a "tag" or a "name" to whatever ultimately matches that portion of the search string.

Pieces of the string that are tagged may be used essentially as variables — more on this subject in an upcoming example.

[A left square bracket marks the start of a "tagged" expression.

The pattern which follows is considered separately from the remainder of the search string.

]a A right square bracket followed immediately by an alphabetic character marks the end of a "tagged" expression.

The portion of the string that matches the "tagged" expression is assigned the one-character "tag" denoted by the alphabetic character which followed the "]" character.

& An ampersand character (&) can be used in the replacement text to recall the characters that were matched by the tagged expression. The ampersand means, literally, "what was matched".

Example:

Consider the following task to perform: assume there are a number of data fields defined in a COBOL record layout and that the definition of these fields has been copied from the DATA DIVISION to the PROCEDURE DIVISION:

```
Full Screen Editor:TT-TSP
     +7-10----+---20----+---30----+---40----+---50----+---60----+---70-2+
 225 :     10  CM-NUMBER                      PICTURE X(8).             :
 226 :     10  CM-STATUS                      PICTURE X.                :
 227 :     10  CM-COMPANY                     PICTURE X(25).            :
 228 :     10  CM-ADDRESS-1                   PICTURE X(25).            :
 229 :     10  CM-ADDRESS-2                   PICTURE X(25).            :
 230 :     10  CM-ADDRESS-3                   PICTURE X(25).            :
 231 :     10  CM-POSTAL                      PICTURE X(7).             :
 232 :     10  CM-TELEPHONE                   PICTURE X(10).            :
 233 :     10  CM-TELEX                       PICTURE X(8).             :
 234 :     10  CM-PO-NUMBER                   PICTURE X(16).            :
 235 :     10  CM-ATTN                        PICTURE X(25).            :
 236 :     10  CM-DP-MGR                      PICTURE X(25).            :
 237 :     10  CM-MACHINE                     PICTURE X(8).             :
 238 :     10  CM-MEMORY                      PICTURE X(4).             :
 239 :     10  CM-DISK                        PICTURE X(8).             :
 240 :     10  CM-TAPE                        PICTURE X(8).             :
 241 :     10  CM-NO-TERMINALS                PICTURE X(3).             :
 [_] +7-10----+---20----+---30----+---40----+---50----+---60----+---70-2+
▶Enter Cmd:      Start line:         End line:         After line:      :
Text :                                                                  :
     :                                                                  :
 [_]  Lines: 822  Lang:C  Case:U  Patterns:Y  Seq:Y  Module:TIP/TT-TSP,S
```

The intention is to smash all of the data field descriptions to statements that initialize the fields to spaces.

One (brute force) approach would be to move the cursor to the upper part of the FSE display and personally remove the "PIC ...." part of each statement (this is an instance where a destructive space bar is handy!) and then use a substitute command to change the string "10..." to the string "MOVE 0 TO ".

An alternative approach (which assumes that the user has first used the SE command to turn pattern matching ON) involves only a single substitute command:

```
▶Enter Cmd: SU   Start line:          End line:          After line:      :
Text :'##  *[%.*]Q .*".'                                                   :
     :MOVE SPACES TO Q.                                    :
 [_]  Lines: 822  Lang:C  Case:U  Patterns:Y  Seq:Y  Module:TIP/TT-TSP,S
```

After the substitute command shown above, the lines appear as follows:

```
Full Screen Editor:TT-TSP
     +7-10----+---20----+---30----+---40----+---50----+---60----+---70-2+
 225 :      MOVE SPACES TO CM-NUMBER.                                     :
 226 :      MOVE SPACES TO CM-STATUS.                                     :
 227 :      MOVE SPACES TO CM-COMPANY.                                    :
 228 :      MOVE SPACES TO CM-ADDRESS-1.                                  :
 229 :      MOVE SPACES TO CM-ADDRESS-2.                                  :
 230 :      MOVE SPACES TO CM-ADDRESS-3.                                  :
 231 :      MOVE SPACES TO CM-POSTAL.                                     :
 232 :      MOVE SPACES TO CM-TELEPHONE.                                  :
 233 :      MOVE SPACES TO CM-TELEX.                                      :
 234 :      MOVE SPACES TO CM-PO-NUMBER.                                  :
 235 :      MOVE SPACES TO CM-ATTN.                                       :
 236 :      MOVE SPACES TO CM-DP-MGR.                                     :
 237 :      MOVE SPACES TO CM-MACHINE.                                    :
 238 :      MOVE SPACES TO CM-MEMORY.                                     :
 239 :      MOVE SPACES TO CM-DISK.                                       :
 240 :      MOVE SPACES TO CM-TAPE.                                       :
 241 :      MOVE SPACES TO CM-NO-TERMINAL.                                :
 [_] +7-10----+---20----+---30----+---40----+---50----+---60----+---70-2+
▶Enter Cmd:      Start line:          End line:          After line:      :
Text :                                                                    :
     :                                                                    :
 [_]  Lines: 822  Lang:C  Case:U  Patterns:Y  Seq:Y  Module:TIP/TT-TSP,S
```

In the above example, note the important appearance of the space after the [%.*]Q expression. The space character is required (in this case) to inform FSE how to determine where the tagged expression ends (in this case the tagged expression is the field name).

3.39. GO — Restart Paused Process

The GO program may by used to reactivate a TIP/30 program which was previously PAUSEd (either a foreground or background program).

When a process is "paused", the TIP/30 scheduler simply ignores that process when it is searching for work to do.

Paused processes are revealed by the WHOSON transaction — the terminal name is suffixed with "/Go" to indicate that the process is awaiting a "GO".

Syntax:

① `GO/identifier`

② `GO identifier`

Where:

identifier Specifies the userid or terminal name to be activated.
Prefix notation may be used; eg: GO *BACK$

Example:

```
GO *JANET
```

This command causes all processes running on behalf of userid "JANET" to be reactivated ("JANET" is too long to be a valid terminal name and would therefore only match on userid).

Error Conditions:

No matching user or terminal can be found.

Additional Considerations:

There is no ill effect associated with issuing a GO to a process which is not paused (other than wasting one's time).

3.40. GROUPS — Modify Elective User Groups

The GROUPS transaction may be used to alter the elective groups to which the user belongs. This effect is temporary — the alteration lasts until it is changed again or until the user is logged off the TIP system (either by choice or otherwise).

The "standard order of search" that TIP/30 employs to resolve transaction and file references is:

- userid
- elective groups
- the system group "TIPY".

The GROUPS transaction allows the user to re-specify some of the elective groups.

Syntax:

```
GROUPS[,Q]  g1,g2,g3,g4,g5,g6,g7,g8
```

Where:

Q Command line option that may be used to suppress the screen format that is usually displayed (see examples which follow).

This is the "quiet" option. Most often used when the GROUPS transaction is called by a transaction program.

g1...g8 Up to 8 positional parameters representing desired alterations to the elective groups to which the user belongs.

Note that only 8 parameters are allowed although there may be as many as 16 elective groups (the number depends on a TIP/30 generation option).

The 8 parameters correspond to the user's first 8 elective groups.

Although 8 parameters are allowed, the system may have been configured to allow a maximum number of elective groups that is less than 8. In that case, the system configuration parameter (NumGRPS=) dictates the number of parameters that the GROUPS transaction actually processes.

If the first parameter is an asterisk (*), the GROUPS program resets the user's elective groups to the groups that are set at logon time (according to the GROUPS= or LOGONSET= specification in the user's catalogue record) and ignores all other parameters.

If a parameter is omitted, the implication is taken that the corresponding elective group is not to be changed.

If a parameter contains exactly the character string NONE, the implication is that the corresponding elective group is to be cleared to spaces.

If a parameter contains a character string other than NONE, the name will be validated according to the user's groupset and logonset members and the corresponding elective group is eligible to change.

Additional Considerations:

If any of the group names supplied are not valid members of the user's logonset or various other groupsets, the GROUPS transaction will not change any elective groups.

GROUPS is, therefore, an "all or nothing" operation — either all of the group names provided are valid or no action is taken.

Example:

```
TIP?▶GROUPS,Q ARC,DOC,PAYROLL
TIP?▶
```

If the "Q" command line option had not been specified, the GROUPS program would have output information in a screen format identical to that employed by the WMI transaction program:

```
                                        13:47 THURSDAY SEPTEMBER 7 1989

            User-id: ALLINSON
             Groups:     ARC          DOC          PAYROLL



           Security:    1
     Account number:
           Terminal: PC01 - SPC (24,80)
   Tip Control Area: TIPTCA
          Site name: A.R.C.
     TIP/30 Version: 4.0 C40R0-000
       ICAM Network: NET1  LOCAP: TIP1
       OS/3 version: 12 00 B

System attributes:
GLOBAL ICAM, DMS, DDP, SYSTEM DEBUG.




TIP?▶
```

Error Conditions:

The GROUPS transaction will report errors by displaying a message on the terminal (unless the "Q" command line option is specified). Programs which intend to invoke the GROUPS transaction (via TIPSUB for example) normally set the CDA-OPTION field to "Q" before issuing the call to transfer control.

The GROUPS transaction clears the first 64 bytes of the CDA to spaces (the 8 parameter areas) if the requested changes were NOT completed; otherwise, the CDA parameters will be set to the user's first 8 elective groups as set by the call to GROUPS.

3.41. HANGUP — Hangup Dial Line

The HANGUP program may be used to "hang up" the telephone line on a dial line connection. This program will log off the user (if the user is logged on TIP/30!) and will issue a line release request to ICAM to terminate the telephone connection. The line will normally be configured in ICAM to immediately return to unattended auto answer state.

The HANGUP transaction is intended to be used in a dedicated ICAM environment — HANGUP does not operate in a GLOBAL ICAM.

Syntax:

```
HANGUP
```

Where:

No parameters required.

Error Conditions:

If the user was logged on TIP/30 and could not successfully proceed to the LOGOFF program, an error may be reported.

3.42. HELP — Display Help Information

The HELP program is a utility that displays help information for a specified online program. The user may ask to see the help information for utility programs supplied with TIP/30.

Help information may also be provided (by the installation administrator) for the installation's user programs (or any other item of interest).

Syntax:

```
HELP  [name] [,page#] [,libname] [,prefix]
```

Where:

name The "name" parameter is used to identify the name of the program or item for which help is requested. If omitted, the HELP program displays a menu listing all items for which help is available.

page# This parameter identifies which page of help information is to be displayed first. The default is the first page. If a page number is specified that is out of range (including a non-numeric value), the HELP program displays the first page of help information.

libname This parameter may be specified to identify the logical file name (LFN) of an online library that the HELP program is to use to find the help information. If this parameter is not supplied, the HELP program first attempts to locate the help information in a library with an LFD of "HELP"; if an appropriate item is not found, the search proceeds to the "TIP" library. See also "Additional Considerations" below.

If this parameter is supplied only this library is searched for help information.

prefix This parameter can be specified to override the default library element name prefix that the HELP program uses to construct the library element name to locate.

If this parameter is omitted, the default prefix is "TH$". See also "Additional Considerations" below.

Only the first 3 characters of this parameter are relevant.

Example:

```
HELP VTOC
```

Displays the supplied help information for the program "VTOC".

Error Conditions:

The requested help information may not be available.

Additional Considerations:

Help information is stored as a library element with the name of the element formed as follows: "TH$xxxxx" (the prefix TH$ may be overridden on the command line). The last five characters of the element name are the first five characters of the name the user is expected to key in as command line parameter 1. For example: TH$VTOC for VTOC; TH$SPL for SPL.

The HELP program first attempts to read this element from a library with the logical file name "HELP". If the element (or file) does not exist, a second read is attempted from the "TIP" library.

The user may therefore establish his own library containing help information by creating elements of the correct name in a library which has a catalogued logical file name of "HELP".

The user should avoid modifying the TIP library directly since that library is completely rebuilt when a new release of TIP/30 is installed (and any alterations would be lost).

3.42.1. Call Another Help Module

The CALL directive is used to display help that is subordinate to the help currently being displayed.

Syntax:

```
/CALL    code,suffix[,security]
```

Where:

code The 1 to 8 character value that will be keyed on the HELP screen.

suffix The 1 to 5 character suffix that will be used to generate a HELP module name (ie. TH$suffi).

security The minimum security required to see the help defined by this directive. Default is 255.

Additional Considerations:

/CALL causes HELP to TIPSUB to itself.

3.42.2. Chain to Another Help Module

The CHAIN directive is used to display help that is equivalent to the help currently being displayed.

Syntax:

```
/CHAIN    code,suffix[,security]
```

Where:

code The 1 to 8 character value that will be keyed on the HELP screen.

suffix The 1 to 5 character suffix that will be used to generate a HELP module name (ie. TH$suffi).

security The minimum security required to see the help defined by this directive. Default is 255.

Additional Considerations:

/CHAIN causes HELP to TIPXCTL to itself.

3.42.3. Display Another Help Screen Full

End the current screen full (max 22 lines). Any remaining data will be displayed on subsequent screens.

Syntax:

```
/EJECT
```

3.42.4. Define a Module Title

Supply data for the Title line of the HELP system's screen format (max 60 characters).

Syntax:

```
/TITLE  -------- 60 character title ----------
```

3.42.5. Define Help Module Security

Define the security level required to examine a HELP module.

Syntax:

```
/SECUR [security]
```

Where:

security The minimum security required to see this help. Default is 255.

3.42.6. Sorted Call/Chain Table

Indicate that the Call/Chain table is sorted in ascending order (HELP can find an entry in the table more quickly).

Syntax:

```
/SORTED
```

3.42.7. Display Call/Chain Table

Display all the entries in the Call/Chain table as a 6, 7 or 8 up list on the terminal.

Syntax:

```
/TABLE  [number][,skip]
```

Where:

number The number of entries to be displayed on each line. Acceptable range is 6 through 8. Default is 8.

skip The number of screen lines to skip before the table is displayed (default is 1).

3.43. HO — OS/3 HOLD Command

The HO transaction implements the OS/3 "HOLD" console operator command. The HO transaction is actually a clone of the more powerful SYM transaction. The SYM transaction program can be invoked using an alias name that is interpreted as a specific command. When the SYM program is invoked with a transaction name of "HO", the SYM program assumes that the OS/3 command is "HO".

The OS/3 HOLD command syntax is documented in the operation guide for your system.

The HO transaction requires the same syntax as the console command. The command is passed exactly as entered to the OS/3 command processor for execution as if it had been entered at the system console.

Note: *There is no provision for the return of any completion status.*

Example:

```
▶HO SPL,PR,JO=COBCOMP3
```

The example command holds the spool output for job "COBCOMP3".

3.44. IDA — Interactive Debug Aid

IDA is an online utility program that permits a programmer to interactively debug TIP/30 online transaction programs. When the user activates IDA on behalf of a program, IDA is given control by TIP/30 and then executes the user program one instruction at a time using the hardware execute (EX) instruction.

After each user program instruction has been "executed" (by IDA), the results and effects of that execution may be displayed on the terminal in a format similar to the assembly language representation of the instruction.

The information presented includes:

- program and job region relative address
- condition code setting
- instruction mnemonic and operands
- effective addresses (operands 1 and 2)
- first four bytes of operands 1 and 2.

During program tracing and execution, the user has a private copy of the load module of the program (regardless of the declared usage of the program in the TIP/30 catalogue).

The programmer using IDA is free to make alterations to the memory regions assigned to the program (the CDA, WORKarea etc), to registers and to instructions themselves. IDA is not available to users who have a security level below programmer level.

WARNING

Extreme caution must be exercised when making alterations with IDA. Indiscriminate alteration of memory (especially memory that is not assigned to the traced program!) can cause the TIP/30 system to crash without warning.

To debug a program that is called directly from the TIP/30 command line, enter a question mark ("?") character preceding the transaction code on the command line:

```
TIP?▶?PAYUPD
```

This technique is preferred when the program to be debugged is the first (or only one) to be called.

Programs that are called from other programs or via IMS succession should be debugged by altering the catalogue entry for the program (this eliminates the tedium of tracing through all of the preliminary programs!).

In order to have a program loaded with IDA, catalogue the transaction code with DEBUG=IDA. Now, whether invoked directly or by succession, when the transaction is loaded IDA is also loaded and given control first.

When IDA receives control, it displays information about the current environment (transaction name, load module name) and prompts the terminal user for a debugging command.

To begin debugging simply press the [XMIT] key. IDA begins executing the program at the address specified as the entry address by the Linkage Editor. At any time the user may interrupt the display by pressing [MSG WAIT] or a function key. IDA pauses and allows the user to enter any of the IDA commands described in the following section.

IDA can be extremely useful even to programmers who are not that familiar with assembly language programming (although a knowledge of assembler is invaluable).

Most high-level languages provide at least a VERB level or source statement address map (in COBOL-74 or COBOL-85 this can be obtained by specifying the compiler parameter PROVER=YES).

The COBOL Data Division map that describes the layout of various areas of the program's assigned storage areas can be very useful too.

3.44.1. IDA COMMANDS

IDA commands are detailed in this section.

An important point: the commands are all one or two characters. Commands that are letters may be entered in upper or lower case (IDA is only case sensitive when it is looking at a character string that is enclosed in quotes).

Unless specifically indicated otherwise, most of the values are entered in hexadecimal (that is, 40 means X'40' — 64 decimal).

When a general purpose register is to be identified, IDA recognizes decimal values from 0 through 15 inclusive or A through F inclusive (to mean registers 10 through 15).

There are 4 floating point registers named 0, 2, 4 and 6.

In general, IDA is not very forgiving about typing errors; in particular, imbedded spaces in hex or decimal values are not permitted. See some of the examples which appear later for clarification of this point.

IDA always issues a prompt when it is expecting a command. If the prompt is suffixed by the string "(LOCK)", IDA is reminding the programmer that some serial resource is locked by the program (for example, a file is in sequential mode).

Although IDA (for obvious reasons) must leave the resource locked while waiting for keyboard input, you are being warned that you are potentially tying up that serial resource and may be delaying other users who need access to that serial resource. The long and short of it is: hurry up!

blank No command — resume program tracing.

+,offset:len

Display storage.

Display next "len" or 16 bytes from the address last used in a D or A command.

IDA remembers the last address which was displayed or altered.

The user may specify a hexadecimal offset to be added to the address.

If an offset is not specified, a default value of 16 is used.

16 bytes will be displayed by default.

A hex length may be supplied (eg: +,50:5A) which results in the display of more (or less) than the default 16 bytes.

-,offset Display previous storage.

Similar to "+,offset" but interprets any specified offset as a negative value.

A pra,n Alter storage.

Beginning at the program relative address (given as PRA) store the value specified as 'n'.

The value may be specified as a hexadecimal string or a character string within single quotes.

AA addr,n Alter storage at absolute address.

Similar to "A" command except that the "addr" is specified as an absolute address rather than a program relative address.

AR r,n Alter general purpose register.

"r" is a decimal number or a letter from "A" through "F" (inclusive) specifying the register to change.

B pra,m,c Specify breakpoint address.

"pra" is the program relative address at which the user wishes to interrupt execution of the program to be able to use other IDA commands (to display or alter memory for example).

A maximum of 16 breakpoints are allowed.

This command is usually followed by the DISPLAY OFF command ("F") when the user wishes to inhibit the IDA display until a specific address has been reached.

"m" is the display mode option to be executed when the breakpoint occurs but before prompting (usually "C", "I", or "N").

"c" is the decimal count of the number of times the breakpoint address is to be encountered before stopping.

When a breakpoint has been reached its address is displayed as well as information describing the display status (on/off) and the current execution mode (Continuous or Instruction).

BG pra,m,c

Set a breakpoint and immediately resume tracing or execution. This command is identical to the "B" command with the exception that IDA does not prompt for another command, but continues execution of the program. In effect, the

"BG" command combines a "B" command with an implied subsequent null command.

BZ pra,m

Breakpoint when content of byte changes. The byte at the specified location ("pra") is monitored by IDA. If the value in the byte changes, a breakpoint is taken.

When the breakpoint is taken, the mode specified as "m" is set (typically, the mode is specified as "I" — instruction step mode).

This command is especially handy to catch situations where a byte value (a switch perhaps?) is being altered incorrectly.

C Continuous Mode display.

When in this mode (the default mode when IDA is first activated), IDA traces the execution of the program until the user interrupts by pressing MSG WAIT or some other function key.

The display rolls up as each instruction displayed.

C+ op1,op2

Hex arithmetic!

The character immediately following the "C" must be one of + - * / % denoting, respectively: addition, subtraction, multiplication, division and remainder.

The operator is assumed to be between the two supplied hex values (op1 and op2).

The two operands must be separated by a comma.

Eg: C+ 3e4,1afb = 0001edf

/ gives the quotient after division

% gives the remainder after division

D pra:len

Display storage.

Display storage in hex and graphics starting at the user program relative address given as PRA.

len is a hex count of bytes to display; the default is 16 bytes.

DA addr:len

Display storage from absolute address.

Display storage in hex and graphics starting at the absolute address given as "addr".

len is a hex count of bytes to display; the default is sixteen bytes.

DB Display Breakpoint Table.

Displays current breakpoint addresses and their associated options and counts.

DE pra Display storage (edited).

Treat the specified program relative address as if it is an instruction and display that location as an readable instruction.

DF Display floating point registers.

Displays floating point registers 0, 2, 4, and 6.

Each floating point register is 64 bits wide but is displayed as left and right 32 bits.

DI r,offset:len

Display indirect using register contents.

Display storage in hex and graphics from the address computed as the sum of the contents of register "r" and hex offset "offset".

If "len" is not specified, 16 bytes will be displayed.

Otherwise "len" specifies a hex number of bytes to display.

DR r Display General Purpose Register(s).

Display the contents of the specified register (either a decimal number from 0 through 15 or a hex character A through F).

If "r" is omitted, all 16 registers are displayed.

E End tracing.

Tracing of the program is discontinued. The user program continues executing in normal mode (not executed via IDA).

If a subsequent program check occurs, control will revert to IDA (for further debugging).

ED End tracing with IDA and dump if subsequent program check.

Similar to the "E" command except that a subsequent program check will pass control to PMDA instead of IDA.

F Display OFF.

Turn the IDA display off.

Usually used to inhibit the tedious display of instructions while waiting to reach a breakpoint that has been set.

G pra GO TO address.

Alter the PSW to execute the next instruction at the specified program relative address given as "pra".

I Enter single instruction mode display.

The user is prompted for an IDA command after every instruction; forces display mode ON.

L addr Specify link address.

The user may enter a specific address which IDA is now to consider as relative location zero.

For example, the address of the program's work area or CDA may be specified so that subsequent use can be made of offset addresses from that base.

To reset the link address to the start of the program, simply enter an address of zero on an "L" command.

As a convenience to programmers using high-level languages, the following reserved words are accepted and will be computed by IDA:

L PIB — base address is the PIB area

L CDA — base address is the CDA area

L MCS — base address is the MCS area

L WRK — base address is the work area

L IMA — base address is the IMA (for IMS programs)

L OMA — base address is the OMA (for IMS programs)

L VOL — base address is the VOLatile area

Master level users are also able to link to the address of the GDA by using the command: "L GDA".

Example of using L command:

```
▸L wrk
```

After issuing this command, relative location zero is considered to be the start of the program's WORKarea.

If, for example, the COBOL Data Division map indicates that a particular field in the work area has a displacement of 1E3 bytes from the start of the work area, the following command could be used to display that (relative location) in the WORKarea:

```
▸d 1e3
```

To see more than 16 bytes (the default length for the D command) enter the following (40 hex is decimal 64):

```
▸d 1e3:40
```

N Display ON.

Turn the IDA display on.

Often used as a breakpoint option.

O,pra Omit Breakpoint.

The program relative address specified by "pra" is removed (omitted) from the breakpoint table.

O* Omit all Breakpoints.

All breakpoints that have been set are cancelled and removed from the breakpoint table.

Q Stop program execution.

Both IDA and the traced program are immediately terminated and control returns to the previous program on the stack.

S pra,n Search for specific hex value or character string.

Search for the value "n" (up to 4 bytes) from the program relative address "pra".

The search argument may be specified as a hexadecimal string or as a character string enclosed in single quotes.

If the search argument is omitted, IDA will continue searching (for the last search argument specified) from the current location.

T mnemonic

Translate mnemonic to opcode.

Translate an assembler mnemonic for an instruction to its internal hexadecimal representation.

Used when one cannot remember the opcode for a assembler instruction.

Example: T MVC returns D2

TN hexop

Translate opcode to mnemonic.

Translate the specified hex opcode to its equivalent assembler mnemonic.

Example: TN D5 returns CLC

3.44.2. IDA Command Examples

The following table summarizes the syntax of IDA commands by illustrating some examples.

Table 3-10. IDA Command Examples

Command	Description
(blank)	Continue executing traced program.
L 128	Link address zero relative to 128.
D 23A	Display 16 bytes at program relative address 23A.
DR	Display all 16 general purpose registers.
DR 11	Display general purpose register 11.
DF	Display all 4 floating point registers.
DI 12	Display 16 bytes at address in general purpose register 12.
DI C,40	Display 16 bytes at address in R12 + X'40'.
B D6,n,10	Stop at program relative address D6 on 10th encounter and execute IDA command "n" — display on.
O D6	Omit breakpoint previously set at D6.
B 7A	Stop at program relative address 7A.
DB	Display entries in breakpoint table.
F	Disable IDA trace information display.
N	Enable IDA trace information display.
I	Set IDA into single instruction display mode.
C	Set IDA into continuous display mode.
C+ 1E45,2EC	Compute: 1E45 + 2EC (hex).
C* 2E,C	Compute: 2E * C (hex).
A 23A,D200F002E000	Alter memory at 23A to X'D200F002E000'.
A 23A,'T301'	Alter memory at 23A to C'T301'.
AR E,1A0	Alter register E (R14) to contain 1A0.
S 24A,47B0F00E	Search for 47B0F00E from address 24A.
S	Search for next occurrence of previous search argument.
S 45E	Search from 45E for previous search argument.

continued ...

Command	Description
S 0,C1C2	Search for C1C2 from location zero.
T CLC	Translate "CLC" to hex opcode.
TN 41	Translate X'41' to mnemonic opcode.
G 128	Go to program relative address 128.
DA 2364A	Display absolute location 2364A.
DE 128	Display instruction at 128.
ED	End tracing (allow PMDA dump).
E	End tracing allow user program to execute (recall IDA if subsequent program check).
Q	Cancel IDA session; end traced program.

3.44.3. IDA Example

```
TIP?▶?calendar 7 1976
'TIP/30 Interactive Debug Aid'
    Transaction code = CALENDAR,   LOAD MODULE = XT$CAL00
Program address = 0C0340 Relative location zero = 000698
IDA:CALENDAR(1)?▶l cda
Program address = 0C0340 Relative location zero = 005100
IDA:CALENDAR(1)?▶d 8
▶  000008 F0F0F0F0 F1F9F7F6 40404040 40404040 - '00001976        '
IDA:CALENDAR(1)?▶a 8,'00001776'
▶  000008 F0F0F0F0 F1F7F7F6 40404040 40404040 - '00001776        '
IDA:CALENDAR(1)?▶e
```

In this example, the user has invoked the CALENDAR transaction with the second command line parameter deliberately set to an incorrect value.

Once IDA is invoked and running, IDA displays the actual program load address and the address that is currently considered to be location zero.

The programmer issues the command l cda to set relative address zero to the start of the CDA.

Using the start of the CDA as the assumed base address, the d 8 command displays 16 bytes of data (the first command line parameter is in the first eight bytes of the CDA; the second parameter starts at offset 8).

The alter command a 8,'00001776' changes 8 bytes at location 8 (of the CDA) to the character string 00001776. Of course, one could have been more succinct and issued the command a d,'7' and altered the one offending byte only!

IDA is then terminated by using the E command (this discontinues IDA and lets the program run without tracing).

3.45. ILLTRN — Illegal Transaction Handler

ILLTRN is a reserved transaction name that may be defined in the TIP/30 Catalogue to process undefined transaction codes that are entered on the TIP/30 command line.

When a user keys a transaction name on the TIP/30 command line and a spelling mistake is made, the TIP/30 system normally reports:

```
Invalid transaction code: xxxxxxxx
```

An installation administrator may prefer to have a user-written program handle invalid transaction codes.

To accomplish this a transaction named ILLTRN must be defined. The presence of a definition of ILLTRN in the TIP/30 catalogue causes the TIP/30 command line processor (TCP) to schedule the ILLTRN transaction if the transaction name that is entered is invalid.

For example, one could take the approach that a misspelled transaction code implies that the user needs some assistance and that the best thing to do is to invoke the TIP/30 HELP processor.

This is accomplished by defining the ILLTRN transaction in this way:

```
TIP?▶CAT
CAT(1)?▶prog tip$Y$/illtrn from=tip$Y$/help.
 PROG catalogue record added: TIP$Y$/ILLTRN
CAT(1)?▶end
TIP?▶
```

Having done this, any user who enters an invalid transaction code finds the TIP/30 HELP program automatically loaded.

Of course, defining ILLTRN in the group TIPY has an impact on all users of the TIP/30 system (since all users are implicitly members of the group TIPY). It is more prudent to define ILLTRN appropriately for individual groups of users.

Additional Considerations:

The ILLTRN processing described here only applies to invalid transaction codes that are entered at the terminal keyboard in response to the standard TIP/30 system prompt that is issued by the TIP/30 Command Line Processor (TCP).

Programs which attempt to internally transfer control to a misspelled transaction receive PIB-NOT-FOUND status.

3.46. IVP — Installation Verification

The IVP program verifies that various files required for the operation of TIP/30 are present and accessible. Information is displayed on the terminal showing the LFD and LBL names of the files.

Syntax:

```
IVP
```

Where:

No parameters are required.

Example Output of IVP:

```
TIP?▶ivp
The following TIP/30 files are open:
    Lfd name  File label
    ========  ==========
    TIP$SWAP  TIP.SWAP
    TIP$CAT   TIP.CAT
    TIP$MCS   TIP.MCS
    TIP$RNDM  TIP.RNDM
    TIP$B4    TIP.B4
    TIP$JRN   TIP.JRN
TIP/30 Installation verification complete.
TIP?▶
```

3.47. JBQ — Display OS/3 Job Queue

The JBQ program is a utility program that displays information about the OS/3 job queue. It is similar to the OS/3 operator command "DI JBQ".

The JBQ program recognizes the following commands:

Command	Description
A	Display all queues.
E	End the JBQ program.
HE	Display command help information on the terminal.
J	Display reason job is not able to be scheduled.
H	Display the high priority job queue.
L	Display the low priority job queue (OS/3 release 8.2 and later).
N	Display the normal priority job queue.
P	Display the preemptive priority job queue.
Q	End the JBQ program and logoff TIP/30.

The JBQ program may be executed interactively or may be given a single command via the command line. If a single command is given on the command line, JBQ will attempt that one command and then terminate normally.

If used interactively, JBQ will prompt the user for each command until an "End" or "Quit" command is given.

3.47.1. Display All OS/3 Job Queues

This command causes the JBQ program to display the status of all the OS/3 job queues: Normal priority, High, Preemptive and Low (if configured).

All jobs in each queue will be shown; job names shown in parentheses are currently on hold.

Syntax:

```
A
```

Where:

No parameters required.

3.47.2. End JBQ Program

This command causes the JBQ program to stop prompting the user for further commands and terminate normally.

Syntax:

```
E
```

Where:

No parameters required.

3.47.3. Display Help Information

This command causes the JBQ program to display help information on the terminal. The help information is a summarization of the recognized command syntax.

Syntax:

```
HE
```

Where:

No parameters are required.

Additional Considerations:

"HE" is the shortest possible string of characters that may be entered for this command.

3.47.4. Display High Priority Queue

This command displays jobs that are in the OS/3 high priority job queue. Jobs currently held by the OS/3 operator are displayed with the job name in parentheses.

Syntax:

```
H
```

Where:

No parameters required.

3.47.5. Display Job Status

This command displays the reason a queued job is not able to be scheduled (eg: device not available, no memory, etc). The specified job name must be a job that is currently queued.

Syntax:

```
J  jobname
```

Where:

jobname The name of the job that cannot be scheduled.

Example:

```
TIP?▶js nitejob
Job NITEJOB not scheduled: Waiting until 89/200 at 23:30
TIP?▶
```

In the above example, the job NITEJOB is awaiting deferred execution at 23:30 on the Julian date 89/200.

Additional Considerations:

The clone transaction "JS" may be used as a shorthand notation for running JBQ with the J command.

Eg: JS TJ$COB74

3.47.6. Display Low Priority Queue

This command displays the jobs that are in the Low priority job queue (available on OS/3 release 8.2 and later). Job names shown in parentheses are currently on hold.

Syntax:

```
L
```

Where:

No parameters required.

Error Conditions:

The Low queue may not be configured or available on the release of OS/3 that is running.

3.47.7. Display Normal Priority Queue

This command displays the jobs in the normal priority job queue.

Syntax:

```
N
```

Where:

No parameters required.

3.47.8. Display Preemptive Priority Queue

This command will display jobs in the preemptive priority job queue. If the system was generated without preemptive job scheduling the display will indicate no jobs in that queue.

Syntax:

```
P
```

Where:

No parameters required.

3.47.9. End JBQ Program and Logoff

This command causes the JBQ program to stop prompting the user for further commands and terminate. If the JBQ program is executing at program stack level one (ie: was NOT called by another program) the user will be logged off the TIP/30 system.

Syntax:

```
Q
```

Where:

No parameters required.

3.48. JCL — Job Submission Utility

The JCL program allows the user to enter job control statements at the terminal to be submitted directly to the OS/3 run processor. This eliminates the necessity of creating an element in a library for quick one-time-only jobs.

The JCL program first calls the transaction code "JCLEDT" to allow the user to create (or modify) a job control stream. This transaction code normally invokes the standard TIP/30 Full Screen Editor (FSE) but may be (individually) tailored to call any available editor.

Once the editing is complete, the user may either End or Quit the editor. If the END option is chosen, the JCL program submits the contents of the edit buffer to the OS/3 run processor via the input reader queue and leaves the edit buffer intact (for possible later modification).

If the QUIT option is chosen, the JCL program prompts the user to determine whether or not to submit the edit buffer before scratching the edit buffer. In this way, the confident user can submit the edit buffer and coincidently discard the buffer.

Syntax:

① `JCL [file/elt [,type]] [,buffer]`

② `JCL [buffer]`

Where:

file/elt Optional file and element to initially read into the editor's buffer.

type The type of element to read — default is source ("S").

buffer The name of the edit buffer that will be accessed. If an edit buffer of that name does not already exist, the JCL program will create it.

If the name is not specified, it defaults to "JCL$tttt" where "tttt" is the ICAM name of the submitting terminal.

Example:

```
JCL TEST
```

Accesses or, if necessary, creates the edit buffer named GROUP1/TEST (where GROUP1 is the name of the first elective group to which the user belongs). The "JCLEDT" transaction is called and, if the user exits the editor with the "E" command, the buffer will be submitted to the OS/3 reader.

3.49. JI — Execute System Command

The JI transaction is used to submit a "console command" to the OS/3 operating system. Examples of "console commands" include use of the RU and RV commands. This transaction code is provided to provide compatibility with the command of the same name that is available with IMS transaction processing systems.

Syntax:

```
JI  cmd
```

Where:

cmd The console command text that is to be submitted to the operating system.

Example:

```
JI RV TJ$COB74:TIPJCS,,E=PAY040
```

This example submits the illustrated RV command to the operating system. The following message is displayed on the screen if the command was accepted by the operating system (*accepted — not necessarily processed successfully!*)

```
JOB REQUEST NOW UNDER SYSTEM CONTROL
```

If the command is not accepted, the following message is displayed on the terminal:

```
INVALID COMMAND  ****  TEXT=
...command text...
```

3.50. JS — Display Job Status

The JS transaction displays the reason why a queued job is not able to be scheduled (eg: device not available, no memory etc). The specified job name must be a job that is currently queued and awaiting execution (otherwise, the JS transaction will report "Job not found" — implying that the named job is not queued).

The JS transaction is a clone of the more general transaction named "JBQ" (refer to "3.47.5. Display Job Status" on page 3-173). When the JBQ program detects that it has been called by the transaction name "JS", it assumes that the desired JBQ function is "J" (job status) and performs that one function and terminates.

Syntax:

```
JS  jobname
```

Example:

```
TIP?▶js nitejob
Job NITEJOB not scheduled: Waiting until 89/200 at 23:30
TIP?▶
```

In the above example, the job NITEJOB is awaiting deferred execution at 23:30 on the Julian date 89/200.

3.51. LC — List TIP/30 Catalogue Information

The LC transaction is a clone of the CAT transaction program. When the CAT program is invoked with the transaction code "LC", it reacts by assuming the desired CAT command is "List" and uses the command line parameters accordingly.

For additional information, see the description of the CAT program in "3.11. CAT — TIP/30 Catalogue Manager" on page 3-11 (specifically the "List" command).

Syntax:

```
LC  p1,p2,p3,p4
```

Where:

The four possible command line parameters are the same parameters that may be specified to the CAT program "List" command.

Example:

```
LC  EDP * * *
```

This command lists all TIP/30 catalogue entries in the group "EDP".

3.52. LIST — LIST Utility

The LIST transaction is a clone transaction of the generalized librarian utility transaction TLIB (see "3.90. TLIB — Librarian Services" on page 3-325) .

The LIST transaction invokes the TLIB program. When the TLIB program observes that the transaction name is not TLIB, it uses the transaction name as the implied command.

The command line options and parameters that are supplied with the LIST transaction code are interpreted by TLIB as parameters to the TLIB LIST command.

The end result is the ability to use LIST as an apparently stand-alone transaction.

Syntax:

```
LIST[,options]  parameters
```

Where:

options Any command line options (as recognized by TLIB) that pertain to the LIST command. See description of TLIB options in "3.90.2. TLIB Options" on page 3-327 and see description of options that affect the TLIB LIST command in "3.90.11. LIST — List Input at Terminal" on page 3-341 .

parameters Parameters required by the LIST command of the TLIB program.

Example:

```
LIST  SRC/BUDGET,S
```

This example lists a source element named BUDGET in the OS/3 library defined with a logical file name of SRC. The output is listed on the terminal with automatic continuation prompts after each screen of data.

3.53. LOGOFF — Log off TIP/30 System

The LOGOFF program is used to log off the TIP/30 system. Only users that are logged on the TIP/30 system may log off — specific terminals *may be* designated as LOGON=NO in the TIP/30 Generation parameters.

The LOGOFF program displays the following screen format that shows information concerning the TIP/30 session just terminated: user-id, site name, number of terminal input and output messages, number of file I/O operations, total time logged on, and the average response time.

```
ALLINSON
Logged off at 11:56:26 on 89/07/26 Site: ARC-TORONTO
        Messages  Input:          3
                  Output:         5
              File I/O's:         0

    Duration of session:    0:00:26
  Average response time:      0.072 seconds
```

The count of the number of I/O operations includes data files, libraries, dynamic files and edit buffers but DOES NOT include other TIP/30 system files (such as TIP$SWAP, TIP$CAT etc).

Syntax:

```
LOGOFF
```

Where:

No parameters are required.

Error Conditions:

An attempt to logoff is not allowed at a stack level higher than the base level (stack level 1).

Additional Considerations:

If the TIP/30 logoff program is invoked via a transaction code that begins with the character "S" (the SOFF transaction for example), TIP automatically issues a "$$SOFF" command to Global ICAM.

Another clone name for the LOGOFF program is FIN.

3.54. LOGON — Log on TIP/30 System

To be able to use the TIP/30 system, the user is required to identify himself to TIP/30. To do this, the user must execute the LOGON program. The LOGON program requires the user to supply his userid and current logon password. The user may be required to supply an (optional) account number (this requirement is installation dependent).

The userid, password and account number supplied by the user are validated according to information in the TIP/30 catalogue.

Syntax:

```
LOGON  [userid]  [,password]  [,account]
```

Where:

userid Userid (maximum 8 characters) assigned to the user by the installation administrator.

password Current password (maximum 8 characters) associated with this userid.

Alphabetic characters in the password may be entered in upper or lower case; the LOGON program automatically converts alphabetic characters to upper case.

account A four character accounting code to associate with this session. The account number is defined by the installation administrator and may be optional.

For additional information, refer to the description of the USER definition keywords DFLTACCT= and ACCT= in "3.11. CAT — TIP/30 Catalogue Manager" on page 3-11.

Example:

```
LOGON FRED/QWERTYUI
```

Logs on a user named "FRED" who has a current password of "QWERTYUI".

Additional Considerations:

If no command line parameters are given or there is any error in the command line parameters, the LOGON program displays the following screen format to assist the user to LOGON correctly:

```
TIP/30   LOGON    Friday July 13, 1984          Time: 11:07

              Site: -site--name-

                    Please logon

          User-id : ________
         Password :
   Account Number : ____

  Place cursor here  (_)  and press XMIT
```

If the TIP/30 system cannot locate the TIP/30 logon screen format (as might be the case when TIP/30 is initially being installed), the following prompt is issued instead of the screen format shown above.

```
THURSDAY JULY 13 1989              Please logon

Allinson-Ross Corporation     TIP/30    Logon
Enter:User-Id/Password/Account-No    Site = xxxxxxxxxxx
 ▶
```

The user should enter his userid, password and (if required) his account number. A maximum of five logon attempts are allowed.

Error Conditions:

The userid may not be valid, the password may not match the current password for the userid, or the account number may be invalid (not in the list of valid accounts for the userid).

Additional Considerations:

If the user does not press XMIT (transmit) within 60 seconds, the LOGON program clears the screen and terminates. The user may reenter the LOGON program to attempt to logon again.

If the user fails to logon after five attempts, the LOGON program imposes a penalty — any LOGON attempt from that terminal is not allowed for 5 minutes.

The following message is displayed at the terminal:

```
You are not authorized to use the system

  Operations control has been notified of
        your attempt to log on
```

and the following message is displayed at the console:

TI058 Unauthorized user attempted logon at ____

Subsequent attempts to logon (during the 5 minute penalty period) result in the following message on the terminal:

```
                Too many attempts to LOGON

            This terminal has been temporarily disabled
```

Note: *The 5 minute penalty may be rescinded by the system operator by using the following unsolicited console command:*

```
EXEC SET FOR xxxx ENABLE
```

The system may be unable to accept LOGONS at the present time (usually this is because the console operator has issued a command to inhibit logons). This condition is usually temporary. The user receives the following message if this is the case:

```
            LOGON requests are not allowed at this time.
```

The user may be prohibited (by the system administrator) from being logged on more than a certain number of terminals at the same time. In this case, the LOGON program displays the following message:

```
Too many users with this User-id are logged on.
  You may not access the system at this time.
```

The userid may have expired (the catalogue record specified the EXPIRY= keyword). The following message is displayed:

```
Your User-id has expired!
System access denied.
```

If the terminal has been set in "test mode", the LOGON program issues a prompt to ask the user if test mode is to remain set:

```
Should terminal be left in test mode? ▶No  ▶Yes
```

3.55. MEM — OS/3 Memory Map

The MEM program is a utility which displays the current OS/3 memory utilization (map). The program details job name, memory region in hex, size in decimal, type, executing, program name, CPU time, account number, storage protect key, executing priority and scheduling priority.

Syntax:

```
MEM[/B]  [ W ]  [dest]  [wait]
```

Where:

B Command line option to include operating system buffer pool information in the display.

W Optional parameter to cause the MEM program to periodically refresh the display on the screen until a function key or the (MSG WAIT) key is pressed. The rate at which the information is refreshed is controlled by the parameter following the "W".

dest This parameter is used to specify a standard TIPPRINT output destination for the output of the MEM program.

The defaults are:

- "AUX0" (full screen) for interactive users
- "ROLL" (line by line output to the terminal) if the "W" command is specified
- "PRNTR" for background users.

wait This parameter is used in conjunction with the "W" specification in parameter one. A numeric value from 5 to 60 (inclusive) may be specified. The value is taken as the number of seconds in the refresh interval.

Default value is 10 seconds (this value is also used if the value specified is not numeric or is not within the acceptable range).

Example:

```
MEM W
```

This command invokes the MEM program to display the current OS/3 memory usage map. Because the firsp parameter is "W", the MEM program runs continuously and refreshes the screen display every 10 seconds until the program is terminated.

Example of MEM Display:

```
                    ..........OS/3 Memory Map..........
  Name      Address      Size    Type      Program    Step   CPU     Acct Key  Pri
SY$STD00 000000-03C4FF   241k Supervisor
SL$$VT00 03DE00-03E8FF     2k Symbiont                          .0  (VR)  00  00
RC$$IS00 03E900-0476FF    35k Symbiont                         1.9  (IS)  00  00
SL$TCA00 04A200-04C9FF    10k Symbiont                          .0  (TW)  00  00
SL$DMS00 06EE00-0711FF     9k Symbiont                          .0  (DM)  00  05
SL@@CM00 07BA00-07E5FF    11k Symbiont                          .0  (CM)  00  01
ML$$C200 07E600-122AFF   657k Symbiont                          .0  (C2)  00  00
         136000-1399FF    14k Free
         13B000-2F6FFF  1776k Free
TIP32B   2F7000-3ADFFF   732k Batch Job  TB$TIP00     1       34.0  TIP3  07  02,H
TIP40T   3AE000-461FFF   720k Batch Job  TB$TIP00     2       29.3  TIP   05  01,P
DMS      462000-4EFFFF   568k Batch Job  DBMS0000     1        4.8  DBMS  04  06,N
TIPTST   4F0000-5E7FFF   992k Batch Job  TB$TIP00     2      101.3  TIP3  03  02,H
TIPDEV   5E8000-762FFF  1516k Batch Job  TB$TIP00     2      421.3  TIP3  02  01,H
GUST     763000-768FFF    24k Batch Job  ML$$GI00     1         .1  GUST  01  04,H




TIP?
```

Additional Considerations:

To discontinue the memory display with the wait parameter, press a function key or the MSG WAIT key.

3.56. MODE — Specify Mode of Operation

The MODE program is used to set debug mode for a terminal or change the terminal display mode to roll mode or scroll mode.

Syntax:

```
MODE[,opt]
```

Where:

opt Option field that determines the action of the MODE program.

D The presence of a "D" character anywhere in the option field places the terminal in debug mode.

In debug mode, the TIP/30 Data Management interface does not perform file operations that would result in alteration of data. This applies to ADD, DELETE, UPDATE operations. TIP/30 returns "good" status to the program (as if the update was performed).

Debug mode does not apply to TIP/30 Edit Buffers or Dynamic files, but does apply to OS/3 libraries.

S The presence of an "S" character anywhere in the option field puts the terminal in scroll mode.

In scroll mode, line oriented output to the terminal will scroll downward from the top of the screen display and wrap around to the first line from the bottom.

Additional Considerations:

When a terminal is set in debug mode, the standard system prompt "TIP?▸" is replaced by "Test Mode: TIP?▸" to serve as a reminder that debug mode is active.

The option field may contain either or both the characters "D" or "S" (in either order). If a "D" is not present, Debug mode will be turned off. Similarly, if an "S" is not present, the terminal will return to ROLL mode.

Example:

```
MODE,S          - scroll mode; set off debug mode
MODE,D          - set on debug mode; roll mode
MODE,DS         - scroll mode; debug mode
```

3.57. MSG — Send Terminal a Message

The MSG program allows a terminal user to send a one line message to a logged on TIP/30 user, a specific terminal, the computer system operator, or to all TIP/30 users who are currently using a specific file (LFD name).

If the destination is not valid, the sender receives an error message. The message text is restricted to a maximum of 64 characters.

If the message is directed to the computer operator, it is displayed on the system console (TIP/30 will not wait for a reply or acknowledgement that the message has been seen!).

If the message is directed to a TIP/30 terminal, the message is sent as an unsolicited message (the recipient will observe the Message Waiting alarm on his terminal and a continuous beep).

The message is displayed on the screen when [MSG WAIT] is pressed. The message is displayed wherever the cursor happens to be resting when [MSG WAIT] is pressed! It is a good idea to first move the cursor to an area of the screen that is not is use.

Syntax:

```
MSG[/dest]  text
```

Where:

dest The desired destination for the message.

May be a userid, terminal name or program name (all users who match will receive the message).

Prefix notation may be used if the destination is a userid, terminal name or program name.

If the destination is specified without prefix notation, the MSG utility will also attempt to match users who are using an filename (LFD) that matches the destination.

If the destination is omitted, the message will be sent to the system console.

text The text to be sent (maximum of 64 characters).

The text does not have to be enclosed in quotes.

If quotes are included, they are treated as text.

The text of the message is translated to upper case before the text is sent.

Example:

```
MSG/BETTY INVENTORY UPDATE IS COMPLETE.
MSG/TRM1  you can try to log on now!
MSG   HOW LONG WILL THE SYSTEM BE UP?
MSG/MANUFIL   MANUFACTURING FILES BEING CLOSED SOON!
```

Error Conditions:

The MSG program reports "Nothing found." if the program cannot send the message to at least one destination.

Additional Considerations:

A message that is sent to the OS/3 operator is split into two output lines if the text is too long for one line on the console.

In a Global ICAM environment, a message may be sent to a terminal only if that terminal is session connected to the TIP/30 locap (via $$SON or a SESSION statement in ICAM). An attempt to send a message to a terminal that is not currently session connected to the TIP/30 locap is reported as "Nothing found.".

3.58. MSGAR — Message Archiver

MSGAR is a utility program that provides librarian services for TIP/30 screen formats (sometimes called "messages"). Screen formats are stored in the TIP$MCS file and may also be pooled in memory for fast access (refer to the TIP/30 generation parameter MCSPOOL=).

TIP/30 screen formats are referenced by an 8 character name. A screen format also has an associated "group" name. The type of group searching that takes place for a screen format depends on the setting of the MCSEARCH= keyword in the user catalogue record.

Syntax:

```
MSGAR[,group]  [command [params] ]
```

Where:

group The name of the group that MSGAR initially considers the "operating" group. This group name becomes the implied screen format group for the MSGAR commands which are entered.

The default operating group is TIPY.

The MSGAR "Group" command may be used at any time during an interaction with MSGAR to alter the current "operating" group.

command A summary of valid MSGAR commands follows.

params Any parameters that are required by the command.

Example:

```
MSGAR,EDP
```

Sets the "operating" group to "EDP". Subsequent MSGAR commands operate on screen formats in the group "EDP".

Additional Considerations:

If MSGAR is invoked with command line parameters, it attempts only that command and then terminates normally.

If MSGAR is invoked without parameters (ie. interactively) it prompts the terminal for each command.

The MSGAR utility recognizes the following commands:

Table 3-11. MSGAR Commands

Command	Description
ALTROF	Turn OFF PIB-ALT-ROW-MCS capability for a screen format.
ALTRON	Turn ON PIB-ALT-ROW-MCS capability for a screen format.
COBol	Create COBOL copy element from a screen format.
COPy	Make a copy of existing screen format in another group.
CUrsor	Change cursor resting location for a screen format.
DATE>	Select only formats with date greater than a supplied date.
DATE<	Select only formats with date less than a supplied date.
DATE=	Select only formats with date equal to a supplied date.
DELete	Delete a screen format.
DIrectory	Print a directory of screen format names and information.
End	End MSGAR program.
FSLOF	Do NOT allow fields to span lines of a screen format.
FSLON	Allow fields to span lines of a screen format.
Group	Change operating group name.
Help	Display help information on terminal.
List	List screen format names and information.
Move	Move existing screen format to different group.
Print	Print a hard copy image of a screen format.
Quit	End MSGAR program and logoff.
REName	Rename a screen format.
REStore	Restore a screen format from an OS/3 library element.

continued ...

Command	Description
RM	Restore multiple screen formats.
RPGIn	Create RPG II Input specifications.
RPGOut	Create RPG II Output specifications.
RPGIND	Change RPG II indicator character for screen format.
Save	Save a screen format in an OS/3 library element.
SR	Save (regardless) — ignore potential library element overwrite prompt.
SM	Save multiple screen formats.
SRM	Save (regardless) multiple screen formats — ignore potential library element overwrite prompt.
SEOF	Do NOT allow Special Emphasis = NOSEON.
SEON	Allow Special Emphasis = NOSEOF.
TABOF	Turn auto-tabbing OFF.
TABON	Turn auto-tabbing ON.
Test	Invoke MSGSHOW to test screen format.
UNIOF	Turn unidirectional field flag OFF.
UNION	Turn unidirectional field flag ON.
Write	Create a library element with group and names list.

3.58.1. ALTRxx — Toggle Alternate Row

This command enables or disables the ability to allow a screen format to be displayed at an alternate row via the PIB-ALT-MCS-ROW field.

Syntax:

```
ALTRON    *name
ALTROF    *name
```

Where:

ALTRON Enable the ability to use alternate rows.

ALTROF Disable the ability to use alternate rows.

***name** The name of a single screen format or a prefix specification to process several formats.

3.58.2. COBOL — Create Cobol Copy Element

This command creates a COBOL copy element from the selected screen format. The copy element identifies each data field with the name "FILLER" and gives the correct picture clause according to the definition of the field in the screen format. A comment line is generated as the last statement indicating the sum of the sizes of all error fields.

Syntax:

```
COBol  *name [,lib  [,elt] ]
```

Where:

***name** The name of a single screen format or a prefix specification to process several formats.

file The logical file name of the OS/3 library

elt The name of the element in the library which is created containing the COBOL statements. The element name defaults to the name of the screen format that was specified.

If screen formats are being processed by prefix the element name must be omitted.

Example:

```
COBOL TESTMSG,PRODSRC/TESTCOB
```

Create a COBOL copy element in OS/3 library catalogued with the logical name of PRODSRC and the element name TESTCOB.

3.58.3. COPY — Copy Screen Format

This command makes a copy of an existing screen format. The "copy" may be given a new format name and may be placed in a different group than the original format. The original format is not affected in any way by this command.

The existing screen format is assumed to be from the current operating group.

Syntax:

```
COPy   name [,newgrp] [,newname]
```

Where:

name The name of an existing screen format (prefix specification is not allowed).

newgrp Optional group name for the copy.

Default is the current operating group.

newname The desired new name of the copy format. This name must be supplied if the [newgrp] is the same as the current operating group.

Default is the name of the existing screen format.

Example:

```
COPy  TF$LOGON,EDP
```

Create a copy of the TF$LOGON screen from the current operating group. The copy of the screen has the same format name (TF$LOGON) but is in the group "EDP".

3.58.4. CURSOR — Specify Cursor Location

This command alters the cursor resting location for a screen format. The cursor location is specified as a row and column number (relative to 1) or may be omitted. If the row and column are omitted, the archiver computes the resting location as the first position of the first unprotected field. If there are no unprotected fields in the screen format, the cursor is placed in column 1 of row 1.

Syntax:

```
CUrsor      name      [,row,column]
```

Where:

name The name of a single screen format (prefix specification not allowed)

[,row,column]

Cursor resting location (home position is 1,1).

Example:

```
cursor testmsg,5,51
```

Force the cursor resting location for screen format named "TESTMSG" (in the current operating group) to row 5 column 51.

Error Conditions:

The specified screen format may not be found or the row or the column specification may be incomplete or invalid.

3.58.5. DATE — Select by Date

This command causes succeeding MSGAR commands to operate on a subset of the available screen formats.

Syntax:

```
DATE<yymmdd
DATE>yymmdd
DATE=yymmdd
```

Where:

DATE<	Select formats whose date is less than "yymmdd".
DATE>	Select formats whose date is greater than "yymmdd".
DATE=	Select formats whose date is equal to "yymmdd".
yymmdd	The date to be used for subset selection.

3.58.6. DELETE — Delete Screen Format

This command deletes a specified screen format from the current operating group.

In order to minimize the impact of fumble-finger typists, the screen format name for this command may NOT be specified using prefix notation.

Syntax:

```
DELete      name
```

Where:

name	The name of a single screen format (prefix specification not allowed)

Example:

```
DEL testmsg
```

Delete the screen format named "TESTMSG".

Error Conditions:

The specified screen format may not be found.

3.58.7. DIR — Directory of Screen Formats

This command produces a printout containing information known about the selected screen formats. The information printed includes: screen name, author, date and time created, total data field count, etc.

Syntax:

```
DIrectry      *name      [,printer]
```

Where:

***name** A single screen format name or a prefix specification

printer The output printer destination. The default destination is PRNTR (the site printer). The printer may also be specified as an auxiliary print device.

Example:

```
DIR    !test,aux1
```

Produce a directory listing of all screen formats which have a name NOT starting with the string "TEST". The printout is to be directed to the auxiliary printer for the issuing terminal.

3.58.8. END — End MSGAR Program

This command causes the message archiver to terminate processing normally.

Syntax:

```
End
```

Where:

No parameters are required.

3.58.9. FSL — Toggle Fields Span Lines

This command enables or disables the ability to allow a screen format to support fields that span lines.

Syntax:

```
FSLON    *name
FSLOF    *name
```

Where:

FSLON To enable the ability to span lines.

FSLOF To disable the ability to span lines.

***name** The name of a single screen format or a prefix specification to process several formats.

3.58.10. GROUP — Specify Operating Group

This command alters the current operating group for the MSGAR program. The current operating group is used as the default group for other MSGAR commands. If a new group name is not specified with this command, the Group command displays the current operating group that is in effect.

Syntax:

```
Group          [grpname]
```

Where:

grpname The desired new operating group.

If this parameter is not supplied, the current operating group is not changed but merely reported at the terminal.

Example:

```
TIP?▶msgar
MSGAR(1)?▶Operating on MCS format group: TIP$Y$
MSGAR(1)?▶g edp
MSGAR(1)?▶Operating on MCS format group: EDP
MSGAR(1)?▶del foo
MSGAR(1)?▶Screen format EDP/FOO deleted.
MSGAR(1)?▶e
TIP?▶
```

3.58.11. HELP — Command Help

This command causes the message archiver to display a summary of recognized commands and required parameter syntax.

Syntax:

```
Help
```

3.58.12. LIST — List Format Summary

This command displays (on the terminal) a summary listing of information known about the selected screen formats. The information is similar to that shown by the DIRECTORY command.

Syntax:

```
List       *name
```

Where:

***name** A single screen format name or a prefix specification

Example:

```
LIST   *test
```

Produce a listing of all screen formats which have a name starting with the string "TEST".

3.58.13. MOVE — Move Screen Format

This command moves an existing screen format (from the current operating group) to a new group. An optional new format name may be specified at the same time to rename the format as the move is taking place.

The original existing screen format (in the current operating group) no longer exists after this command is used.

Syntax:

```
Move    name,newgrp   [,newname]
```

Where:

name The name of an existing screen format (prefix specification is not allowed).

newgrp New group to move the screen format to.

newname Optional new name for the screen format.

Default is the name of the existing screen format.

Example:

```
Move  TEST,TIP$Y$,ORDER001
```

Move the screen format "TEST" from the current operating group to the group TIPY as screen name "ORDER001".

3.58.14. PRINT — Print Screen Format

This command creates a hard copy image of specified screen formats. The image is a representation of the screen as it was defined to the TIP/30 Format Definition program (see "3.89. TFD — Screen Format Definition" on page 3-283).

Data fields and heading fields are shown with original edit and control information. The hard copy image may be routed to the site printer PRNTR (the default destination) or to an auxiliary print device (for example: AUX1).

Syntax:

```
Print   *name    [,printer]    [,case]    [,,NB]
```

Where:

***name** A single screen format name or a prefix specification

printer The name of the destination printer (default is PRNTR; other examples are: AUX1 AUX1*BYP). Any printer name recognized by TIPPRINT may be specified.

case A choice between "Upper" and "Lower" indicating the desired case of the printout.

"Upper" is the default if the destination is the site printer;

"Lower" is the default when the destination is an auxiliary printer.

NB If this parameter is specified, the Print command does not attempt to print data field codes and special heading fields in "bold" (multi-strike).

Example:

```
PRINT  *test,,LOWER
```

Produce a hard copy printout of all screen formats with a name starting with the string "TEST" on the site printer and attempt to print lower case data.

3.58.15. QUIT — End MSGAR Program

This command ends the message archiver. In addition, if the user was executing the message archiver at stack level 1 (ie: the message archiver was NOT called from another program) then the user is logged off the TIP/30 system.

Syntax:

```
Quit
```

3.58.16. RENAME — Rename Screen Format

This command renames an existing screen format. The new name must be a name that is not currently in use.

Syntax:

```
REName      name,newname
```

Where:

name The name of an existing screen format

newname The desired new name for the screen format

Example:

```
ren     testmsg,xtestmsg
```

Change the name of screen format "TESTMSG" to "XTESTMSG".

3.58.17. RESTORE — Restore Screen Formats

This command restores screen formats that were previously saved (by the message archiver) in an OS/3 library element. The name of the element containing the saved screen format need not be the same as the name of the screen format.

Syntax:

```
REStore     name      ,file      [,elt]
RM          *name     ,file      ,elt
```

Where:

name The name of a single screen format.

Prefix specification is only allowed with the RM (Restore Multiple) command.

file The logical file name of the OS/3 library

elt The name of the element in the library which contains the saved screen format(s).

Default is 'name' with the Restore command.

Must be supplied with the RM command.

Example:

```
REST    TESTMSG,PRODSRC/XTESTMSG
```

Restore (recreate) a screen format called "TESTMSG" from library "PRODSRC" element "XTESTMSG".

3.58.18. RPG — Create RPG II Layout

This command creates RPG II input or output specifications from the selected screen format.

Each data field in the screen format is defined with a unique name (for that format).

Syntax:

```
RPGIn     *name [,lib] [,elt]
RPGOut    *name [,lib] [,elt]
```

Where:

name The name of a screen format.

Prefix specification is allowed.

file The logical file name of the OS/3 library.

Default is RUN (the TIP job's YRUN library).

elt The name of the element which is to be created.

Default is the name of the screen format that was specified.

If screen formats are being processed by prefix, the element name must be omitted.

Example:

```
RPGOUT  TESTMSG,PRODSRC/TESTCOB
```

Create RPG II output specifications, for the format TESTMSG, in OS/3 library catalogued with the logical name of PRODSRC and the element name TESTCOB.

3.58.19. RPGIND — Change RPG II Format Id

This command changes the format identification character for a screen format.

Any input specifications created via the RPGIn command show this value.

Syntax:

```
RPGIND   *name     [,char]
```

Where:

name The name of a screen format.
Prefix specification is allowed.

char The character to use as the format identification character.
Default is space.

Example:

```
RPGIND   TESTMSG,D
```

Change the format identification character for TESTMSG to a "D".

3.58.20. SAVE — Save Screen Format

This command saves one or more screen formats in an OS/3 library element (or elements if the SM (Save Multiple) or SRM (Save Multiple Regardless) commands are used).

The save command is useful for taking a backup of screen formats before undertaking extensive modifications or in preparation for transporting screen formats to another TIP/30 system.

Syntax:

```
Save      *name     ,file     [,elt]
SR        *name     ,file     [,elt]
SM        *name     ,file     ,elt
SRM       *name     ,file     ,elt
```

Where:

name The name of a screen format.
Prefix specification is allowed.

file The logical file name of an OS/3 library.

elt The name of an element that is to be created (in the library) to contain the screen format(s) selected.

With the Save and SR commands, the element defaults to the name of the screen format that is being saved; if screen formats are selected by prefix the element name must be omitted.

With the SM and SRM commands, the element name must be specified.

Example:

```
SAVE    *TF$,BACKUP
```

Save all screen formats with a name starting with "TF$" into the OS/3 library catalogued with the logical file name "BACKUP". Each element is created with the name of the screen format it contains.

Additional Considerations:

If the command is specified as "SR" (Save Regardless) or SRM (Save Regardless Multiple), the MSGAR program does **not** prompt the user with an overwrite check (if the element already exists). This is a dangerous command and should be used with extreme caution.

The Save and SR commands write each screen format selected into it's own library element (default element name is the same as the screen name).

The SM and SRM commands write all screen formats selected into a single library element (the element name must be supplied).

3.58.21. SE — Toggle Special Emphasis

This command enables or disables the ability to use the special emphasis underscore on UTS/30 or UTS/40 terminals.

Syntax:

```
SEON    *name
SEOF    *name
```

Where:

SEON Enable the special emphasis underscore.

SEOF Disable the special emphasis underscore.

***name** The name of a single screen format or a prefix specification to process several formats.

3.58.22. TABxx — Toggle Auto-Tabbing

This command enables or disables the auto-tabbing feature of a screen format.

Syntax:

```
TABON    *name
TABOF    *name
```

Where:

TABON	Enable the auto-tabbing feature.
TABOF	Disable the auto-tabbing feature.
***name**	The name of a single screen format or a prefix specification to process several formats.

3.58.23. TEST — Test Screen Format

This command invokes the MSGSHOW utility (see description in "3.59. MSGSHOW — Screen Format Testing" on page 3-209). to permit the user to test a screen format. The MSGSHOW program is called with the specified screen format name and an underscore as the default filler character to use.

Syntax:

```
Test  name
```

Where:

name	A single screen format name to use to invoke MSGSHOW.

3.58.24. UNIxx — Toggle Unidirectional Fields

This command enables or disables the ability to use the unidirectional fields in a screen format.

Syntax:

```
UNION    *name
UNIOF    *name
```

Where:

UNION	Enable the ability to use unidirectional fields.
UNIOF	Disable the ability to use unidirectional fields.
*name	The name of a single screen format or a prefix specification to process several formats.

3.58.25. WRITE — Write Screen Format Name List

This command creates an element in an OS/3 library which contains all the specified screen format names. Each "line" of the created element contains a single group/name pair. The write command is especially useful for creating command files for a subsequent run of the message archiver. The element created by the write command can be edited later using the TIP/30 Text Editor — FSE.

Syntax:

```
Write      *name      [,file]    [,elt]
```

Where:

*name	The name of a single screen format or a name prefix specification.
file	The logical file name of the OS/3 library (default is "RUN")
elt	The name of the element to create (default is "MSGAR")

Example:

```
WR      !TF$,RUN/NONTIP
```

Creates an element named "NONTIP" in library "RUN" containing lines of screen format names that do NOT begin with the string "TF$".

3.59. MSGSHOW — Screen Format Testing

This program (via either of two transaction codes) is used to test TIP/30 screen formats:

MSGTST

Prompts the user for test data and presents it on the terminal using the named screen format.

MSGSHOW

Displays a specified screen format (without data), waits for the user to enter test data, then displays the data as the data would be received by a program using the screen format.

In either case, the unformatted data screen expects the test data to be a continuous character string. A user can cycle back and forth between screens trying various data entry options.

When the user's formatted message is displayed, intentional errors may be introduced to check error field options. Entering a circumflex as the first character in a field and pressing the XMIT key causes the field to blink and an error message to be displayed.

MSGTST and MSGSHOW display the data received exactly as it would appear in a user program MCS-DATA area. Note that no header information or communications characters are received, and that the number of characters sent is a function of cursor position.

Numeric fields are returned to the program right justified and zero filled. Data characters entered into a field which are incompatible with the field definition are replaced by blink characters (or are blinked) by the Message Control System. The errors may be corrected and data changed to try out various options available. Simply place the cursor after the data and press the XMIT key.

Syntax:

① `MSGTST[,grp]  format  [,fill]  [,func]`

② `MSGSHOW[,grp]  format  [,fill]  [,func]`

Where:

grp The group name associated with the screen format to be tested.

Default action is to search through all of user's groups.

format The name of the screen format to be tested.

fill The fill character to be used while testing the screen format. Choices are: asterisk, underscore, the word SPACE or NO to indicate a space fill character.

Default: underscore is used as fill character.

func The MCS-FUNCTION code (see description of this field in the MCS section of the TIP/30 Programming Guide). If not specified, MCS-FUNCTION is set to space.

Example:

```
MSGSHOW  TF$FSE03
```

3.60. NEWUSER — Logon as Another User

There is often the necessity to terminate the current session (logoff) and start another session (logon) using a different user-id or account number. To simplify this process, the NEWUSER program is provided.

NEWUSER enables a logged on user to logoff and logon in one step.

Syntax:

```
NEWUSER  userid  [/password]  [/account]
```

Where:

userid	Userid to log on.
password	Current password associated with the userid.
account	The account number to use to LOGON.

Example:

```
NEWUSER FRED/QWERTYUI,A106
```

Logoff the user that is presently logged on and immediately attempt to LOGON as user "FRED" using the current password for FRED (illustrated as the string "QWERTYUI") with an appropriate account number.

Error Conditions:

The userid may not be valid, the password may not match the current password for the userid, or the account number may not be valid for the specified userid. If any of these errors occur, TIP/30 presents the user with the logon screen format (the implied logoff is usually successful).

3.61. NOTE — Display Informational Message

The NOTE program allows a terminal user to send a message to the terminal which invoked the NOTE program. This command is usually used within .IN files to signal progress through the command file (see description of input redirection in the documentation of the TIP/30 Program Control System — PCS).

Syntax:

```
NOTE[,W]    text
```

Where:

W	A command line option character to cause the NOTE program to pause and wait for any input (usually a function key or the [MSG WAIT] key). This option is often used in TIP/30 redirected input files (sometimes called "command files") — see the description of redirected input in the documentation of the TIP/30 Program Control System — PCS. The text is displayed and the NOTE program waits for any input before continuing.
text	The message text (64 characters maximum) that is to be displayed on the terminal.

Example:

```
NOTE    ALL USER-IDS HAVE BEEN CATALOGUED
NOTE,W PRESS MSG-WAIT TO CONTINUE
```

3.62. PAUSE — Pause Executing Process

The PAUSE program may by used to suspend execution of a TIP/30 process (either a foreground or background process).

When a process is "paused", the TIP/30 scheduler ignores that process when it is searching for work to do.

If a paused process is the target of a DIE or PURGE request, an automatic "GO" is issued to allow the process to die.

Paused processes are revealed by the WHOSON transaction — the terminal name appears with the suffix "/Go" to indicate that the process is awaiting a "GO".

Syntax:

① `PAUSE/identifier`

② `PAUSE identifier`

Where:

identifier The userid or terminal name to be paused.

Prefix notation may be used; eg: PAUSE *BACK$

PAUSE will not act on the process that is executing (that is, you may not PAUSE yourself).

Example:

```
PAUSE *JANET
```

This command pauses all processes running on behalf of userid "JANET" ("JANET" is too long to be a valid terminal name and is therefore only a potential match for a user name).

3.63. PMDA — Post Mortem Dump Analysis

PMDA is a dump analysis program that enables a programmer to interactively examine a dump from an on-line program. PMDA is automatically invoked by TIP when a user program aborts. PMDA creates a dynamic file containing a copy of the user program memory areas at the time of the dump.

The dynamic file is created with a name constructed as follows:

```
userid/DUMPtttt/trid
```

where "userid" is the userid of the user executing the program that aborted, "tttt" is the ICAM terminal name of the user terminal, and "trid" is the catalogued transaction name that invoked the program that aborted.

If the user is an application level user, PMDA prints the dump at the site printer (see description of the "P" command) and terminates normally. However, if the user is at programmer level security (or higher), PMDA allows the user to enter commands to "browse" through the dump at the terminal. The programmer level user may specify that the dump file is to be printed and/or kept.

PMDA may be invoked directly from the terminal to browse through a previously kept dump file (see syntax description following).

Another important function of the PMDA program is to release any files that may have been assigned to the program.

PMDA is most often encountered as a result of a program abort condition. However, it is possible to execute PMDA directly as a transaction to continue analysis of a previously retained dump.

To execute PMDA interactively, the command line syntax is:

Syntax:

```
PMDA  trid  [,tttt]  [,userid]
```

Where:

trid	The name of the transaction that aborted.
tttt	The ICAM name of the terminal where the original abort occurred (default is the current terminal).
userid	The userid that was running the program at the time the program aborted (default is the current userid).

PMDA recognizes the following interactive commands:

Table 3-12. PMDA Commands

Command	Description
D	Display areas of memory of the aborted program.
E	End interaction with PMDA (retain dump file).
P	Print hard copy of dump.
Q	End interaction with PMDA (scratch dump file).

Programmers generally elect to print a dump whenever a transaction program aborts. In some cases, it is possible to first browse through the dump at the terminal and discover the reason the program terminated abnormally (and therefore eliminate the need to print the dump).

Some familiarity with assembler programming concepts is assumed in the following discussion of PMDA commands.

3.63.1. D — Display Memory

This command displays the contents of the memory allocated to the program that aborted. The display command has several variations to allow the programmer to specify storage or registers to display.

D addr

Display 16 bytes in hexadecimal and graphic from the specified address.

D ABORT

Display the abort area (the 32 bytes surrounding the instruction that failed). The word "ABORT" must be entered as illustrated.

D name [,offset]

Display 16 bytes in hexadecimal and graphic from the start of the linkage area given by "name" plus optional offset. The recognized names are:

PIB	CDA	MCS	IMA	OMA	WORK

Offset is specified as a hexadecimal value; if omitted, the offset defaults to zero.

DF Display the contents of the floating point registers.

D PSW Display the abort address and the PSW at the time the program aborted.

DR Display the contents of the general purpose registers.

Example:

```
D 5800     - Display 16 bytes from address X'5800'.
D MCS,40   - Display 16 bytes at offset X'40'
             from the start of the MCS area.
```

3.63.2. E — End Program

This command ends interaction with PMDA and retains the dynamic file containing the dump.

Syntax:

```
E
```

Where:

No parameters required.

Additional Considerations:

The TIP/30 dynamic file is created with a name constructed as follows:

```
userid/DUMPtttt/trid
```

"Userid" is the userid of the user executing the program that aborted, "tttt" is the ICAM terminal name of the user terminal, and "trid" is the catalogued transaction name that invoked the program that aborted.

To reload PMDA to resume analysis of the dump, invoke PMDA interactively as described in "PMDA — Post Mortem Dump Analysis" on page 3-214.

3.63.3. P — Print Dump

This command causes PMDA to create a printed dump for offline analysis by the programmer. Unless an optional print destination is specified, PMDA first attempts to print the dump to logical file name "TIP$PMDA", otherwise the dump is printed using the logical file name "PRNTR".

The dump is formatted to assist the programmer: major areas of storage are identified in much the same fashion as an OS/3 SYSDUMP.

When the printed report is completed, PMDA terminates after scratching the temporary dynamic file that contained the memory contents (effectively performing an automatic "Q" command after printing).

Syntax:

```
P [dest]
```

Where:

dest Optional parameter to override the default print destination. Any valid TIPPRINT printer name may be specified. Default is first TIP$PMDA, then PRNTR as described above.

Additional Considerations:

The TIP/30 system administrator may elect to define an additional print file in the TIP/30 generation parameters and job control (LFD PRNTRX for example) and create a catalogue entry for logical file name "TIP$PMDA" so that PMDA printed reports are directed to a separate printer from PRNTR:

TIP/30 Gen:	FILE PRNTRX,PRINT BUFFER=1.

TIP/30 JCL:	// DVC 21
	// SPL HOLD
	// LFD PRNTRX

TIP/30 Catalogue:	FILE TIPY/TIP$PMDA SECUR=255 LFD=PRNTRX.

3.63.4. Q — End and Scratch Dump File

This command causes PMDA to scratch the temporary dump file and end interaction with the user.

Syntax:

Q

Where:

No parameters required.

3.64. POC — Terminal Reset

The POC transaction handles a power-on confidence test (POC) interrupt from a terminal. Some terminals (SVT-112x for example) send notification to the host computer when the terminal is reset. This notification is (and always has been) translated by TIP/30 into the pseudo function key number 23.

The TIP/30 Catalogue defines the POC transaction to invoke the TT$MOD load module. When the TT$MOD program is called, it determines the correct screen size, whether or not FCCs are supported, updates TIP/30 internal terminal tables if necessary, and displays a confirmation message on the terminal.

Each site may define a TIP/30 transaction with the trid "F#23" that has the same characteristics as POC to automatically perform this diagnosis if F#23 is received while the TIP/30 system prompt appears on the terminal.

If a function key (even a pseudo function key like 23) is pressed while the TIP/30 command line processor is active, TCP first checks to see if there is a transaction defined for the user with the reserved name of the function key (eg: F#23). If there is such a transaction defined, it is executed; otherwise, TCP proceeds to check the user's function key definitions (see "3.21. DEFKEY — Define Function Keys" on page 3-73).

The most common use of the POC transaction is to inform TIP/30 that an SVT-112x terminal is being changed between 80 column and 132 column mode. The terminal operator can change the setting of the control page and then invoke the POC transaction directly.

Example of POC Output:

```
▶The terminal size is now set to: 24 X 80 - FCCs are supported.
TIP?▶
```

3.65. PR — OS/3 PR Command

The PR transaction implements a variation of the OS/3 "PR" console operator command (start a burst mode output writer). The PR transaction is actually a clone of the more powerful SYM transaction. The SYM transaction program can be invoked using an alias name that is interpreted as a specific command. When the SYM program is invoked with a transaction name of "PR", the SYM program assumes that the OS/3 command is "PR".

The OS/3 PR command syntax is documented in the

operation guide for your system.

The PR transaction submits the parameters exactly as entered to the OS/3 system as if the command was entered on the system console.

Note: *There is no provision for returning any completion status.*

Example:

```
PR BU,FORM=STAND2
```

The example command begins a burst mode output writer to process Spool subfiles that specify a form name of "STAND2".

3.66. PRINT — PRINT Utility

The PRINT transaction is a clone transaction of the generalized librarian utility transaction TLIB (see "3.90. TLIB — Librarian Services" on page 3-325).

The PRINT transaction invokes the TLIB program. When the TLIB program observes that the transaction name is not TLIB, it uses the transaction name as the implied command.

The command line options and parameters that are supplied with the PRINT transaction code are interpreted by TLIB as parameters to the TLIB PRINT command.

The end result is the ability to use PRINT as an apparently stand-alone transaction.

Syntax:

```
PRINT[,options]  parameters
```

Where:

options Any command line options (as recognized by TLIB) that pertain to the PRINT command. See description of options that affect the PRINT command in "3.90.12. PRINT — Print Input" on page 3-343.

parameters Parameters required by the PRINT command of the TLIB program.

Example:

```
PRINT  SRC/BUDGET,S
```

This example prints a source element named BUDGET in the OS/3 library defined with a logical file name of SRC. No output specification is provided — the default output location is the site printer (PRNTR).

3.67. PURGE — Remove Process

The PURGE program may be used to forcibly remove a TIP/30 process from the system. This action may be necessary because a terminal has been abandoned while a program that will never timeout is running or because a program is in some sort of endless loop.

> **WARNING**
>
> Use of this transaction (or the corresponding operator console command) should be used with extreme caution.

Syntax:

① `PURGE/identifier`

② `PURGE identifier`

Where:

identifier The userid or terminal name to be purged.

Prefix notation may be used; eg: PURGE *BACK$

PURGE does not act on the process that is executing (that is, you may not PURGE yourself).

Example:

```
PURGE *JANET
```

This command causes all processes running on behalf of userid "JANET" to be purged from the system ("JANET" is too long to be a valid terminal name and would therefore only match a userid).

Error Conditions:

PURGE may report "No matching user or terminal can be found" if the identifier does not match any running process.

Additional Considerations:

Purged programs are aborted immediately without generating any sort of termination dump (PMDA). If the program was running at an execution stack higher than 1, all intermediate stack levels are aborted as well.

The PURGE program generates a console message to indicate that a PURGE operation has taken place:

```
TI055 userid(term): INVOKED PURGE/identifier
```

Any roll back of transactions is properly handled.

The corresponding operator console command may prove to be useful when the operator has tired of waiting for users to voluntarily logoff TIP/30.

3.68. RDR — Create RDR Spool File

The RDR program allows the user to create "card image" data at the terminal (or retrieve such data from a library element) and have this data placed in a standard OS/3 Spool reader sub file.

The RDR program first calls the transaction named "RDREDT". This transaction code is normally defined to invoke the standard TIP/30 Full Screen Editor (FSE) but the transaction may be defined to call any available editor. While in the editor, the user may take advantage of all of the capabilities of the editor to create the desired data images. In particular, FSE can manipulate edit buffers that are 96 or 128 characters wide — the FSE "ma" command (set margins) and the data shift commands (<) and (>) may be used to move the editing window to the left or right.

Once editing is complete, the user may either End or Quit the editor to return control to the RDR program. If the End option is chosen, the RDR program creates a reader subfile containing the data that is in the edit buffer. The edit buffer is left intact (for possible later modification).

If the Quit option is chosen, the RDR program prompts the user to determine whether or not to create the spool sub file before scratching the edit buffer. This way, a confident user can create a reader sub file and have the RDR program discard the edit buffer afterward.

Syntax:

```
RDR[,opt]      [ file/elt [,type] ]   [,buffer]
RDR96[,opt]    [ file/elt [,type] ]   [,buffer]
RDR128[,opt]   [ file/elt [,type] ]   [,buffer]
```

Where:

RDR96 Alternate transaction name, to allow creation of 96 character wide reader files. If this transaction name is used, the FSE editor is called (rather than RDREDT). This is because FSE is the only TIP/30 editor which is able to handle line images greater than 80 characters.

RDR128 Alternate transaction name, to allow creation of 128 character wide reader files. If this transaction name is used, the FSE editor is called (rather than RDREDT).

opt Command line options to control various side effects.

H	Create the reader file in HOLD queue.
R	Create the reader file as RETAIN sub file.
	If neither option "H" nor "R" is specified, the reader file will be marked as queued.

file/elt Optional file and element to initially read into the editor's buffer.

type The type of element to read — default is source ("S").

buffer The name of the edit buffer that will be accessed.

If an edit buffer of that name does not exist, RDR will create it.

If the name is not specified, it defaults to "RDR$tttt" where "tttt" is the ICAM name of the submitting terminal.

This name is used as the resulting LBL name of the reader sub file that is created.

Example:

```
RDR TEST
```

This example accesses (or creates) the edit buffer named grp1/TEST (where grp1 represents the name of the first elective group to which the user belongs).

The "RDREDT" editor will be called and, if the user exits the editor with the "E" command, the contents of the buffer will be created as a reader sub file named "TEST".

Reader sub files can be accessed by a batch program by using job control statements such as those described below:

For RDR or RDR96	For RDR128
// DVC 30	// DVC 130,I
	// VOL X(NOV) or RDR128
// LBL xxx	// LBL xxx
// LFD zzz	// LFD zzz

3.69. RE — Display Ready Message

The RE transaction clears the terminal screen and displays the current TIP/30 greeting (ready message) information. The RE transaction is actually a clone of the more powerful SYM transaction. The SYM transaction program can be invoked using an alias name that is interpreted as a specific command. When the SYM program is invoked with a transaction name of "RE", the SYM program reacts by displaying the TIP/30 "ready message".

Syntax:

```
RE
```

Example output of RE:

```
<<< TIP/30 Version 4.0 C40R0-000  Ready for xxxxxxxxxxxx 89/09/01  15:59 >>>

              * TIP/30 *
              ==========
```

Note: *The string "xxxxxxxxxxx" represents the 12 character TIP/30 site name as specified in the TIP/30 generation parameters for the TIP/30 system that is executing.*

The two lines of greeting text is normally provided in the TIP/30 job control run-time options.

The current date and time is shown along with the current revision level of the TIP/30 software.

3.70. RECOVER — RECOVER Element

The RECOVER transaction is a clone transaction of the generalized librarian utility transaction TLIB (see "3.90. TLIB — Librarian Services" on page 3-325).

The RECOVER transaction invokes the TLIB program. When the TLIB program observes that the transaction name is not TLIB, it uses the transaction name as the implied command.

The command line options and parameters that are supplied with the RECOVER transaction code are interpreted by TLIB as parameters to the TLIB RECOVER (alias BACK) command.

The end result is the ability to use RECOVER as an apparently stand-alone transaction.

Syntax:

```
RECOVER[,options]  parameters
```

Where:

options Any command line options (as recognized by TLIB) that pertain to the RECOVER command. See description of TLIB options in "3.90.2. TLIB Options" on page 3-327 and see description of options that affect the RECOVER command in "3.90.15. RECOVER — Activate Previous Version" on page 3-348.

parameters Parameters required by the RECOVER command of the TLIB program.

Example:

```
RECOVER  SRC/BUDGET,S
```

This example attempts to roll back a source element named BUDGET in the OS/3 library defined with a logical file name of SRC. The most recent previous version of that element in that library is made the active version.

Additional Considerations:

The RECOVER transaction marks the current active version of the module as "deleted" and then "reactivates" the most recent previous version. If both steps cannot be performed, no action is taken. Note that before using RECOVER to roll back an accidentally deleted module, a "dummy" active module of the same name must first be written to the library.

To roll back to an earlier version, simply reissue the RECOVER command as many times as needed.

The RECOVER transaction reports the module comment, date and time stamp of the reactivated module.

3.71. RELOAD — Reload Program

The RELOAD program is used to indicate to TIP/30 that it must obtain a "fresh" copy of a load module from the TIP$LOD library (or the TIP$BLK fast load file — if configured). TIP/30 does not necessarily load a load module from the library (or TIP$BLK file) every time the load module is needed.

When an online program is re-linked, the RELOAD transaction may be used to force TIP/30 to "refresh" the load module before using it again.

The load module to be RELOADed may be in use when the RELOAD program is run. In this situation, the RELOAD program informs the user that the load module is in use and the RELOAD remains pending and is performed when all present users of the load module are finished.

It is possible to RELOAD a RESIDENT program (one that is permanently resident in memory — a JCL start up option). The RELOAD program first warns the user that the load module specified is a RESIDENT program and then requests positive confirmation that the RELOAD is to be performed.

If a RESIDENT program is RELOADed, the resident copy in memory is marked unusable (and whatever memory it occupied is wasted until TIP/30 shuts down).

Syntax:

```
RELOAD loadm
```

Where:

loadm The load module name (trailing zeroes need not be entered).

Example:

```
TIP?▶reload tt$mod
TT$MOD00 cleared from loadr table.
TT$MOD00 cleared from reentrant control table.
TT$MOD00 ver: 89/09/15 @ 10:32 (C) A.R.C  TIP/30 4.0          4051 bytes
TIP?▶
```

As the example above illustrates, the date, time stamp and library comment *from the TIP$LOD library information* is shown along with the size of the load module in bytes.

Additional Considerations:

If the program is being used reentrantly then TIP/30 must wait for all current users of the program to stop using it before a new version can be loaded.

If an attempt is made to reload a resident SUBPROGram, RELOAD displays an error message and terminates without taking any action (reloading a resident SUBPROG is not permitted!).

An alternative transaction code (ZZPCH) is provided for compatibility with other software systems.

3.72. RPG — RPG Editor

The RPG editor is an online program which was written to aid programmers in the creation and maintenance of programs written in the RPG II language. Using the RPG editor, a programmer no longer has to worry about aligning fields in the proper columns. RPG has eight screen formats; one for each of the form types used in writing RPG II programs. The user only has to select the appropriate screen and enter the data on titled blank fields. RPG edits and aligns the data as if it was coded on a card.

There are 10 screen formats used in the RPG editor:

- Menu
- Record list
- Control card format
- File descriptor format
- File extension format
- Line counter format
- Telecommunications format
- Input format
- Output format
- Calculation format.

All commands are issued from the menu screen format. The remaining screen formats can be displayed using commands from the menu.

3.72.1. Invoking RPG

The RPG editor is invoked from the TIP/30 command line using this syntax:

Syntax:

① `RPG`

② `RPG  file/elt`

Format ① is used to create a new source program; format ② is used to update an existing element.

Note: *In the event of a system crash the edit buffer can be retrieved by invoking the RPG editor with just the element name: RPG ELT*

3.72.2. RPG Editor Screen

These are the screen formats used by the RPG editor:

```
TIP/30 RPG - EDITOR            COMMAND MENU                CURRENT LINE:     0

  Command: _
  Line Number(s) From: _____  To: _____
  FILE/ELT Name: ________/________
  File Comment:  ____________________
                                                LEAVE CURSOR HERE -- <_>


------------------------------ COMMANDS ------------------------------------

  H   CONTROL CARD FORMAT              .     DISPLAY CURRENT LINE
  F   FILE DESCRITPION FORMAT          $     DISPLAY LAST LINE
  E   FILE EXTENSION FORMAT            P     PRINT LINES (LIST) MAX 15
  L   LINE COUNTER FORMAT              W     WRITE TO FILE
  T   TELECOMMUNICATIONS FORMAT        Q     QUIT (AND SCRATCH EDIT BUFFER)
  I   INPUT FORMAT                     X     EXIT (AND SAVE EDIT BUFFER)
  C   CALCULATION FORMAT               =     DISPLAY FILE/ELT/COMMENT
  O   OUTPUT FORMAT                    R     READ ADDITIONAL ELEMENTS
  *   COMMENTS                         Y     DISPLAY USER INSTRUCTIONS
                                   MSG WAIT  SAME AS 'Q'
```

```
TIP/30 RPG - CONTROL SPECIFICATIONS                      CURRENT LINE:     0

SPEC Type: H

Compilation Mode     (2,3,4, )  _
ERROR ANALYSIS DUMP  (D, )      _
OPERATOR CONTROL     (1, )      _          COL 10-14  _____
GENERATE DEBUG CODE  (1, )      _          COL 16-20  _____
Inverted Print       (D,I,J, )  _          COL 22-25  ____
ALT COLLATING SEQ    (S, )      _          COL 27-30  ____
BINARY SEARCH        (1, )      _          COL 32-39  _______
SIGN HANDLING        (S,I,B, )  _
Forms Alignment      (1, )      _
INDICATOR INIT.      (S, )      _
FILE TRANSLATION     (F, )      _          COL 44-47  ____
SHARED I/O AREA      (1, )      _          COL 49-69  ____________________
CCA NAME                        ____
SUBROUTINE           (A,S, )    _

                                                    Leave cursor here -- <_>
-------------------------------------------------------------------------------
F1: UPDATE RECORD  F2: DISPLAY NEXT RECORD      MSG WAITING: BACK TO MENU
F3: DELETE RECORD  F4: DISPLAY PREVIOUS RECORD  TRANSMIT:    ADD RECORD TO FILE
```

```
TIP/30 RPG - FILE SPECIFICATIONS                          CURRENT LINE:     0

FORM TYPE: F

FILE NAME:                          _______   TYPE:          (I,O,C,U,D)  _
DESIGNATION:      (P,S,R,C,D,T)     _         EOF:           (E, )        _
SEQUENCE:         (A,D, )           _         FORMAT:        (F,V,D)      _
BLOCK LENGTH:                       ____      RECORD LENGTH:              ____
PROCESSING MODE:  (L,R, )           _         KEY OR R.A. LENGTH:         __
R. A. TYPE:       (A,P,I,K,R)       _         ORGANIZATION:  (I,T,D,X,2) _
OVERFLOW IND.:    (OA-OG,OV, )      __        KEY START LOCATION:         ____
EXT OR LINE CODE: (E,L, )           _         DEVICE:                     _______
SYMBOLIC DEVICE:                    ______    LABELS:        (S,N,E,K, )  _
LABELS OR USER EXIT:                ______    ISAM INDEX RES. SIZE:       ______
ADD OR UNORD LOAD: (A,U, )          _         CYL OVERFLOW SPACE (X10)    _
# OF EXTENTS:                       __        TAPE REWIND:   (R,U,N, )    _
FILE CONDITIONERS (U1-U8)           __                                    __


                                                    Leave the cursor here -- <_>
-------------------------------------------------------------------------------
F1: UPDATE RECORD  F2: DISPLAY NEXT RECORD      MSG WAITING: BACK TO MENU
F3: DELETE RECORD  F4: DISPLAY PREVIOUS RECORD  TRANSMIT:    ADD RECORD TO FILE
```

```
TIP/30 RPG - INPUT SPECIFICATIONS                         CURRENT LINE:     0

FORM TYPE: I

FILE NAME:
SEQUENCE: (01-99,AA-ZZ)             __        NUMBER: (1,N, )                _
OPTIONAL: (O, )                     _         RECORD ID: (01-99,H1-H9,L1-L9,LR)  __
RECORD IDENTIFICATION CODES:                  POSITION   NOT   C,Z,D   CHAR

                                                ____      _      _      _
                                                ____      _      _      _
                                                ____      _      _      _

STACKER SELECT: (1-9, )             _         DATA FORMAT: (P,B,L,R, )       _
FIELD:                                        FROM   TO   DEC  NAME

CONTROL LEVEL: (L1-L9, )            __        MATCH OR CHAIN: (M1-M9,C1-C9, )  __
FIELD REC RELATION: (ANY IND)       __
FIELD INDICATORS:  (01-99,H1-H9, )            PLUS   MINUS   ZERO/BLANK

COL 71-74                           ____        __     __       __
                                                    Leave the cursor here -- <_>
-------------------------------------------------------------------------------
F1: UPDATE RECORD  F2: DISPLAY NEXT RECORD      MSG WAITING: BACK TO MENU
F3: DELETE RECORD  F4: DISPLAY PREVIOUS RECORD  TRANSMIT:    ADD RECORD TO FILE
```

```
TIP/30 RPG - OUTPUT SPECIFICATIONS                          CURRENT LINE:     0

FORM TYPE: O

FILE NAME: ________
TYPE: (H,D,T,E)                  _
STACKER SEL/FETCH (0-9,F, )      _
SPACE:  BEFORE:                  _   AFTER:  _
SKIP:   BEFORE:                  __  AFTER:  __
OUTPUT INDICATORS:             NOT  IND
                                 _    __
                                 _    __
                                 _    __
FIELD NAME:                      ______
EDIT CODE:                       _
BLANK AFTER: (B, )               _
FIELD END POSITION:              ____
DATA FORMAT: (P,B,L,R, )         _
CONSTANT OR EDIT WORD:           ____________________________
COL 71-74                        ____                  Leave the cursor here -- <_>
-------------------------------------------------------------------------------
F1: UPDATE RECORD  F2: DISPLAY NEXT RECORD      MSG WAITING: BACK TO MENU
F3: DELETE RECORD  F4: DISPLAY PREVIOUS RECORD  TRANSMIT:    ADD RECORD TO FILE
```

```
TIP/30 RPG - CALCULATION SPECIFICATIONS                     CURRENT LINE:     0

FORM TYPE: C

CONTROL LEVEL: (L0-L9,LR,SR,OR,AN)  __
INDICATORS:   NOT IND
               _   __
               _   __
               _   __
FACTOR1:       _________
OPERATION:     _____
FACTOR2:       _________
RESULT:        ______
LENGTH:        ___
# DECIMALS:    _
HALF ADJ:      _
RESULTING INDICATORS:     HIGH 1>2 +    LOW 1<2 -   EQUAL 1=2 0

                            __            __          __
COMMENTS:      ______________
                                                 Leave the cursor here -- <_>
-------------------------------------------------------------------------------
F1: UPDATE RECORD  F2: DISPLAY NEXT RECORD      MSG WAITING: BACK TO MENU
F3: DELETE RECORD  F4: DISPLAY PREVIOUS RECORD  TRANSMIT:    ADD RECORD TO FILE
```

```
TIP/30 RPG - LINE COUNTER SPECIFICATIONS                CURRENT LINE:     0

FORM TYPE: L
FILE NAME: ________

LINE #      CH. NO.

 1.   ___     __
 2.   ___     __
 3.   ___     __
 4.   ___     __
 5.   ___     __
 6.   ___     __
 7.   ___     __
 8.   ___     __
 9.   ___     __
 10.  ___     __
 11.  ___     __
 12.  ___     __
                                          Leave the cursor here -- <_>
--------------------------------------------------------------------------------
F1: UPDATE RECORD  F2: DISPLAY NEXT RECORD      MSG WAITING: BACK TO MENU
F3: DELETE RECORD  F4: DISPLAY PREVIOUS RECORD  TRANSMIT:    ADD RECORD TO FILE
```

```
TIP/30 RPG - FILE EXTENSION SPECIFICATIONS              CURRENT LINE:     0

FORM TYPE: E

SEQ OR CHAIN FILE: (01-99,AA-ZZ) __        CHAINING FIELD: (C1-C9) __
FILE NAME FROM:                  ________  TO FILE NAME:           ________
TABLE OR ARRAY NAME:             _______   # OF ENTRIES PER REC:   ___
# OF ENTRIES PER TABLE OR ARRAY: _____     LENGTH OF ENTRY:        ___
DATA FORMAT: (P,B,L,R, )         _         # DECIMALS:              _
SEQUENCE: (A,D, )                _         ALT TABLE OR ARRAY:     ______
LENGTH OF ENTRY:                 ___       FORMAT: (P,B,L,R, )     _
# DECIMALS:                      _         SEQUENCE: (A,D, )       _

COMMENTS:          ________________




                                          Leave the cursor here -- <_>
--------------------------------------------------------------------------------
F1: UPDATE RECORD  F2: DISPLAY NEXT RECORD      MSG WAITING: BACK TO MENU
F3: DELETE RECORD  F4: DISPLAY PREVIOUS RECORD  TRANSMITT: ADD RECORD TO FILE
```

```
TIP/30 RPG - TELECOMMUNICATIONS FORMAT                    CURRENT LINE:     0

Form Type: T

FILE NAME                                 _______  COL 14     _
CONFIGURATION (A-L,N,O,Q-R,T-Z)           _
TYPE OF STATION   (R,T)                   _        COL 17-18  __
TRANSPARENCY    (N,Y, )                   _
SWITCHED    (A,B,E,M,S)                   _        COL 21-47  _____________________________
REMOTE TERMINAL                           ____     COL 52     _
PERMANENT ERROR INDICATOR
        (01-99,H1-H9,L1-L9,LR)            __
WAIT TIME                                 ___      COL 58-59  __
LAST FILE    (L, )                        _        COL 61-64  ____
REMOTE DEVICE                             ______
TERMINAL NAME                             ____


  .                                                    Leave cursor here -- <_>
-------------------------------------------------------------------------------
F1: UPDATE RECORD  F2: DISPLAY NEXT RECORD      MSG WAITING: BACK TO MENU
F3: DELETE RECORD  F4: DISPLAY PREVIOUS RECORD  TRANSMIT:    ADD RECORD TO FILE
```

```
TIP/30 RPG - COMMENT SPECIFICATION                        CURRENT LINE:     0

FORM TYPE: *

COMMENT: ____________________________________________________________________

   (NOTE: THE FIRST CHARACTER OF THE COMMENT FIELD MUST BE AN ASTERISK *)









                                                  LEAVE THE CURSOR HERE -- <_>
-------------------------------------------------------------------------------
F1: UPDATE RECORD    F2: DISPLAY NEXT RECORD       MSG WAIT:  RETURN TO MENU
F3: DELETE RECORD    F4: DISPLAY PREVIOUS RECORD   TRANSMIT:  ADD RECORD TO FILE
```

There are five ways to get a record into these displays:

- from the menu by entering . (current line)
- from the menu by entering $ (the last line)
- from the menu by entering P and the line number (specific line)
- from one of the eight format displays by pressing F3 (next record)
- from one of the eight format displays by pressing F4 (previous record).

The current line number is not altered by updates.

3.72.3. Delete Line

To delete a record from the edit buffer it must be displayed on its correct format screen. Once the record is displayed, it can be deleted by pressing the F2 key. The line number of all records following the deleted record are decreased by one. The current line is the record which immediately followed the deleted record.

3.72.4. Add a Line

The addition of records is done from the formatted screen. If the screen displayed at the moment is not the correct format, the user must intercede by returning to the main menu and selecting the correct RPG II form type. Once the data has been entered, press the XMIT key.

The data is then validated. If all fields are valid, the record is added to the edit buffer and becomes the current line.

3.72.5. Update Line

To update a line, display it on the terminal in correct form type screen, make the necessary corrections to the data and press the F1 key. The data fields are then edited. If they are all valid the old record is replaced by the new record and the user is given update confirmation.

If the validation fails, the fields in error are changed to blinking fields and the record is not updated. The user may correct the fields in error and resubmit or request another screen. Any fields which are numeric or blank only are edited by the screen formatter. Any record in the edit buffer can be updated as long as it is in one of the eight format displays.

3.72.6. List Lines

To list part of the edit file: enter 'P' as the command on the menu and the beginning and end line numbers of the lines to be listed. The records are listed as they would appear on cards except that the line numbers and program identification are not shown. A maximum of 15 lines can be viewed at once. After these lines are listed, the next or previous 15 records may be viewed by pressing function key 1 or 2 respectively. The current line is the last line displayed on the terminal.

Note: *If 'P' is entered without a line number, RPG editor assumes that the user wants the current line displayed in its card format.*

3.72.7. Ending the RPG Editor

To terminate the session enter X or Q as the command in the menu. Entering Q scratches the buffer, entering X retains the buffer. Before entering this command the user may wish to save his updated text by writing it to a permanent file using the W command.

3.72.8. Changing the Current Line

To display the current line in its format display, enter . (a period) as the command on the menu.

To display the last line in the edit file enter $ (a dollar sign) as the command on the menu. The record will be displayed in its corresponding format. The last line now becomes the current line.

To determine the current line number enter = (an equal sign). The line number of the current line is displayed.

3.72.9. Saving Text in a Library

To write the contents of the edit file to a library element, enter command "W" and the filename and element name on the menu. A special comment can be inserted on the file header by entering it next to the "comment" (maximum length is 20 characters). If this entry is left blank, the userid is used as the text of the comment.

On a successful write to the library, the editor responds with the number of lines copied. The edit file does not change as a result of the write command. It is important to remember that the RPG editor works only with a copy of what is in the library file. The content of the library file does not change unless a write command is issued and confirmed.

3.73. RU — Run OS/3 Job

To RUN an OS/3 batch job which requires the use of the card reader, the TIP/30 user may use the SYM program and enter the "RU" OS/3 operator command (see "3.86. SYM — Schedule OS/3 Symbiont" on page 3-276).

A more direct approach is the use of the RU program. The RU program is a clone transaction that invokes the SYM program. The SYM program detects that it has been called with a transaction name of "RU" and reacts appropriately.

The RU program expects (on the command line) the parameters that the user would normally give with the OS/3 operator "RU" command. The user should keep in mind that OS/3 limits the length of a console command to a maximum of 60 characters.

Syntax:

```
RU  parameters
```

Where:

parameters Parameters required by the RU command.

Example:

```
RU LOADCARD
```

Runs a job stream named "LOADCARD" from the system job control library.

Error Conditions:

The user may receive a security error if he does not have sufficient security to run the "RU" program.

3.74. RV — Run OS/3 Job

To RUN an OS/3 batch job, the TIP user may use the SYM program and enter the "RV" OS/3 operator command via the SYM program (see "3.86. SYM — Schedule OS/3 Symbiont" on page 3-276).

A more direct approach is the use of the RV program. The RV program is a clone transaction name that invokes the SYM program. The SYM program detects that it has been called with a transaction name of "RV" and reacts appropriately.

The RV program expects (on the command line) the parameters that the user would normally give with the OS/3 operator "RV" command. The user should keep in mind that OS/3 limits the length of a console command to a maximum of 60 characters.

Syntax:

```
RV  parameters
```

Where:

parameters Parameters required by the RV command.

Example:

```
RV TJ$COB74(FRED),,E=TEST010
```

Runs a job stream named "TJ$COB74" from the system job control library and changes the job name to "FRED". The keyword specification assigns a value to the job global "E".

Error Conditions:

The user may receive a security error if he does not have sufficient security to run the "RV" program.

3.75. SC — OS/3 Operator SC Command

To issue the OS/3 operator console command "SC" to schedule a saved job stream, the TIP/30 user may use the SYM program (see section on "SYM") and enter the "SC" OS/3 operator command via the SYM program.

A more direct approach is the use of the SC program. The SC program is a clone transaction name that invokes the SYM program. The SYM program detects that it has been called with the transaction name "SC" and reacts appropriately.

The SC program expects (on the command line) the parameters that the user would normally give with the OS/3 operator "SC" command. The user should keep in mind that OS/3 limits the length of a console command to a maximum of 60 characters.

Syntax:

```
SC  parameters
```

Where:

parameters Parameters required by the SC command.

Example:

```
SC TIP/30
```

Schedule a previously "saved" job stream named "TIP/30".

Error Conditions:

The user may receive a security error if he does not have sufficient security to run the "SC" program.

3.76. SCR — Scratch OS/3 File

The SCR program will scratch an OS/3 file or files. The user supplies either a file name (LBL name) OR the expiration date OR the file name prefix of the file(s) that are to be scratched.

Syntax:

```
SCR  VSN=   [ FIle='...'     ]
            [ DAte=yy,ddd    ]
            [ PREfix='....' ]
```

Where:

VSN= The volume serial number of the disk containing the file or files to be scratched.

This keyword may be omitted IF the FILE= keyword is specified and the named file is already catalogued in the OS/3 catalogue (YCAT).

FIle= The LBL name of the file to be scratched.

The filename must be enclosed in quotes.

DAte= The expiration date (in julian format — yy,ddd).

When this keyword is specified, ALL files on the volume which have an expiration date LESS than OR EQUAL to the specified date will be scratched.

Specifying DATE=99,999 will scratch all files on the volume!!

PREfix= The common prefix of the filename(s) to be scratched.

The prefix MUST be exactly four (4) characters and MUST be enclosed in quotes.

Note: The specifications FILE=, DATE= and PREFIX= are mutually exclusive. Only one of these keywords may be specified.

Example:

```
SCR VSN=REL130 PREFIX='TEST'
```

Scratch all files on volume REL130 that have an LBL name beginning with the four characters "TEST".

Additional Considerations:

A console message will be generated (TI055) to log the fact that the user has used the SCR program to scratch one or more files.

3.77. SCRATCH — Scratch Dynamic File

The SCRATCH transaction is used to erase a TIP/30 dynamic file that is currently assigned to the terminal. The SCRATCH program removes the entry for the dynamic file from the TIP/30 catalogue and releases the space currently used by the dynamic file in the TIP$RNDM file.

Syntax:

```
SCRATCH[,A]  lfn
```

Where:

A — Command line option used to indicate that **all** TIP/30 Dynamic Files assigned to this process (terminal) are to be scratched. Any OS/3 files that are currently assigned are freed by this option (see "3.37. FREE — De-Access a File" on page 3-100). If this option is specified no file names need be supplied.

lfn — The logical file name (LFN) of the file to be scratched.

Example:

```
SCRATCH TEST1
SCRATCH,A
```

Error Conditions:

TIPFCS errors may be reported.

3.78. SET — Alter Process Attributes

The SET program is a TIP/30 system utility that allows the user to change various attributes of his own or other terminal processes. Certain system attributes may also be changed (subject to security considerations described below).

This program is intended to be used by the system programmer.

Syntax:

```
SET  [ FOR tttt ]  attributes
```

Where:

FOR tttt The terminal name associated with the process to be changed.

Default is the terminal running the SET program.

To use this clause, the user must be at least SYST level security. If this clause is specified, it must appear before any other parameters.

attributes One or more of the following statements:

SPC	Change terminal type to Unisys PC (Personal Computer) or a compatible clone.
TTY	Change terminal type to TTY (teletype).
U10	Change terminal type to U10.
U20	Change terminal type to U20.
U200	Change terminal type to U200.
U30	Change terminal type to U30.
U40	Change terminal type to U40.
U400	Change terminal type to U400.
U400F	Change terminal type to U400F (U400 with character protect feature installed).
U60	Change terminal type to U60.
BYPASS=	Specify BYPASS=tttt to change bypass terminal specification in CLUSTER definition to the terminal named "tttt". The terminal that this statement is to apply to (the FOR terminal) must have an existing CLUSTER specification. Specify BYPASS=' ' (a space in quotes) to change the terminal to have NO bypass terminal.
LMON	Turn TIP/30 software line monitor on.
LMOFF	Turn TIP/30 software line monitor off.
LOGON=	YES means terminal MUST logon to TIP/30.

NO means terminal is NOT required to logon to TIP/30.

NOLOGONS
Temporarily prohibit TIP/30 logons system wide.

LOGONS Inverse of NOLOGONS; allow TIP/30 logons system wide.

DISABLE Disable terminal. NO input will be accepted from the terminal.

ENABLE Enable terminal (inverse of DISABLE).

Also allows penalized terminal to attempt TIP/30 logon.

DEBUG ON
Make the system default to using storage protection.

DEBUG OFF
Clear "DEBUG ON". Storage protection only for programs catalogued as DEBUG=YES.

LF=YES Change CLUSTER TIPPRINT line feed option to YES.

LF=NO Change CLUSTER TIPPRINT line feed option to NO.

LF=NULL Change CLUSTER TIPPRINT line feed option to NULL.

PROMPT= Change the TIP/30 prompt system wide.

Refer to documentation of the TIP/30 system generation keyword parameter PROMPT= for more information.

SIZE= Set terminal size to specified rows and columns.

Choices: SIZE=(24,80) or SIZE=(24,132)

TEST ON Set terminal in test mode. File updates ignored.

TEST OFF The terminal is cleared from test mode.

UNSOL= Specify NO to set the terminal to reject incoming unsolicited messages.

Specify YES to set the terminal to accept incoming unsolicited messages.

Example:

```
SET FOR T312 U200 LMOFF LOGON=YES.
SET U20 LOGON=NO.
```

Additional Considerations:

The user must be at least a SYSTem level user to use the "FOR" clause or to specify DEBUG, NOLOGONS or LOGONS attribute.

3.79. SHUTDOWN — Shutdown Processing

The SHUTDOWN transaction is intended to be a mechanism whereby the system administrator can schedule one or more transactions that are to be run when the TIP/30 system is shutdown in an orderly manner (see description of the EOJ transaction or operator console command).

The SHUTDOWN transaction may be specified as the system SHUTDOWN transaction by specifying SHUTDOWN=SHUTDOWN in the TIP/30 generation TIPGEN statement (or via the corresponding keyword in the TIP/30 job control stream).

The SHUTDOWN program first attempts to open a DEFKEY file named: "SHUTDOWN/FUNCTION/KEYS" — if there is no such DEFKEY file, the program attempts to open the library element: "TIP$LOD/SHUTDOWN,S" (the TIP$LOD library was chosen because it is guaranteed to exist and the contents of that library are preserved across TIP/30 releases).

SHUTDOWN "performs" each line of the input stream (whether it is a line of the DEFKEY file or the library element) as if the line was a standard TIP/30 command line.

Since the SHUTDOWN program MUST run in background (and normally is executed at system EOJ time) any transaction that SHUTDOWN performs must itself be capable of running in background.

The SHUTDOWN transaction, and all transactions that are executed as a result of running that transaction, are executed with the pseudonym userid of "CONSOLE". If a user of that name is defined in the TIP/30 Catalogue, the security level and group memberships of that userid take effect. If the userid "CONSOLE" is not defined, the TIP/30 system assumes a security level of 1 (TECH) and GROUPS=ARC (membership in group TIPY is assumed as always).

For more information about defining a "CONSOLE" userid, see the section on TIP/30 Operator Console commands.

For example, assume that the library element TIP$LOD/SHUTDOWN,S contains the following lines:

```
STATUS
CCA
```

The SHUTDOWN program issues a call to TIPSUB (in turn) to the transaction "STATUS" and "CCA". These transactions run in background and accomplish whatever they do in background (in this example, both programs are designed to run in background and generate a statistical report).

A subtle point is that the SHUTDOWN program opens the library element as a ".IN" file (see description of "Redirected input").

This means that any transactions that are "executed" by SHUTDOWN MAY be followed by commands to that transaction. SHUTDOWN will never "see" those lines because those lines are automatically "read" by the individual transaction.

The SHUTDOWN program may also be given a transaction name of "STARTUP" so that one could have both a SHUTDOWN and a STARTUP set of procedures. The names "STARTUP" and "SHUTDOWN" in the foregoing discussion are interchangeable.

Additional Considerations:

The SHUTDOWN program displays a message on the system console that indicates when it is scheduling (or "performing") a particular subordinate transaction.

The TIP$LOD library is normally restricted to WRITE=NO in the TIP/30 catalogue. Before attempting to write to this library, the system administrator may have to adjust the entry in the TIP/30 catalogue to allow the editor to write to TIP$LOD.

If the SHUTDOWN program cannot find the appropriately named DEFKEY file or the appropriate element in the TIP$LOD library, it simply terminates.

3.80. SOFF — Log Off TIP/30 and $$SOFF

The SOFF transaction is used to log off TIP/30 and to issue a $$SOFF command (to sign off in a GLOBAL ICAM environment).

The SOFF transaction is a clone of the TIP/30 LOGOFF program. When the LOGOFF program detects that it has been invoked with a transaction name spelled "SOFF", it performs an ordinary TIP/30 LOGOFF and outputs a "$$SOFF" command to the terminal. The string "$$SOFF" is output at the home position of the terminal and is followed by the necessary control code to cause an "auto-transmit".

In a GLOBAL ICAM environment, the auto-transmitted "$$SOFF" is intercepted by GUST as if it was keyed in by the terminal operator.

Example:

```
SOFF
```

Error Conditions:

An attempt to logoff will not be allowed at a stack level higher than the base level (stack level 1).

3.81. SORT — Sort Edit Buffer

The SORT program is a utility program that sorts the contents of a TIP/30 edit buffer. The edit buffer to be sorted should contain a reasonable number of lines. The sorting technique used is not suitable for a large number of records because the time taken to sort is not a linear function of the number of lines in the buffer!

Data is sorted according to the standard EBCDIC collating sequence (according to the internal binary representation of each character).

This program is utilized by other TIP/30 transaction programs (USERS and FSE for example) and is not often used directly from the TIP/30 command line.

Syntax:

```
SORT   grp,buffer [,begin] [,end] [,col] [,dir] [,pwd]
```

Where:

grp The name of the group to which the edit buffer belongs.

buffer The name of the edit buffer.

begin The line number where sorting is to begin (inclusively).
Default: line 1.

end The line number where sorting is to end (inclusively).
Default: last line of buffer.

col The starting column of the data to use as a sort key.
Default: column 1.

dir The direction of sorting. May be either "A" (ascending) or "D" (descending).
Default: "A" (ascending).

pwd The password associated with the edit buffer (optional and seldom utilized).

Example:

```
SORT EDP/TABLE,,,10
```

This command sorts an edit buffer named "TABLE" in the group "EDP" into ascending sequence according to data starting in column 10.

Additional Considerations:

The sort is not a stable sort; records with identical sort keys will not necessarily remain in their original order.

The comparison of key information is performed using the hardware instruction "CLC". As a consequence, the SORT program may incorrectly interpret data that is in packed or binary format (for example, positive numeric packed fields may have a "C" or "F" sign — the SORT program will sort X'003C' ahead of X'003F' even though both values represent +3).

3.82. SPL — Spool File Enquiry

The SPL program enables the user to examine subfiles in the OS/3 spool queues. A spool subfile may be listed at the terminal, printed at a terminal printer, released to the system output writer or deleted.

The SPL program is able to read subfiles in the OS/3 spool queues. It has no provision for modification of data in the subfile.

The OS/3 spool file is divided into two classes of subfile:

- Held
- Not Held (queued).

Subfiles that are held are the usual (default) target of the SPL program. It is possible to direct SPL to examine subfiles that are not held, but the user should be aware that subfiles are queued only until the OS/3 output writer opens them for processing. There is, therefore, a potential race condition associated with queued files.

The OS/3 spool file is also divided (for each of the two classes described above) into the following queues:

LOG	Job log.
PR	Local print (default queue for SPL).
PU	Local punch.
RDR	Local reader.
RDR96	Local 96 column reader.
SYSLOG	Retained job log (if configured).
RBPIN	Remote reader (if configured).
RBPPR	Remote print (if configured).
RBPPU	Remote punch (if configured).
DDPPR	Distributed processing printer (if configured).
DDPPU	Distributed processing punch (if configured).

There are 22 (2 x 11) combinations of class and queue.

To examine or manipulate a spool subfile entry, the user must always clearly establish both the class (default is HELD) and the queue (default is PR) of the desired subfile.

Syntax:

```
SPL[/opt]  [cmd [queue] [,option] [ kwd= kwd= ...]]
```

Where:

opt Command line option. The specification DEL turns on the automatic delete option for the PRINT command (delete after printing — a dangerous option).

Also see following description of the PRINT command and the DELETE= keyword.

cmd A recognized SPL program command (eg: DELETE, PRINT) as described in the next sections.

queue The OS/3 spool queue to be searched (default is the print queue: PR).

option Optional additional information required by some commands.

kwd= Optional keywords that are used to qualify the selection of subfiles in the specified queue.

The keywords may be specified in any order and may be separated from other keywords by a comma or one or more spaces.

Additional Considerations:

The SPL program normally operates interactively; that is, the user simply keys in the transaction code (SPL) and subsequently provides commands to perform SPL functions.

The SPL program accepts a single command (with whatever keywords are appropriate) on the TIP/30 command line. In that case, SPL attempts only that single command and then terminates normally.

Example:

```
SPL PRINT JOB=MYCOB74 USING=*BYP,AUX1
```

3.82.1. SPL Command Summary

The following table lists the commands that the SPL program recognizes and provides a brief description of the use of the command.

Table 3-13. SPL Command Summary

Command	Description
DEL	Delete subfile.
E	Terminate SPL program.
H	Display command and operational help information.
L	List subfile on terminal.
LE	List compiler output subfile on terminal — go directly to "error" page.
P	Print subfile.
PC	Print subfile (with compression).
PT	Print subfile with test page.
Q	Terminate SPL program and logoff TIP/30.
R	Release held subfile.
S	Display summary of subfiles.
ST	Display spooling statistics and options.
W	Write subfile to TIP/30 edit buffer.
WL	Write subfile to library element.
WS	Write summary information to library element RUN/SPOOL.

3.82.2. SPL Security Considerations

To maintain the security of the OS/3 spool file, the SPL program displays information from the spool queues according to the following rules:

- MASTER level users (ie: security 1 thru 9) are able to examine any spool queue subfile;
- SYSTEM and PROGRAMMER users (ie: security 10 thru 29) are able to examine any spool queue entry with form name "STAND1";
- Other spool subfiles can be examined by a user if and only if:
  - the TIP/30 userid, elective group, or terminal name matches one of the FORM=, CART=, REMOTE=, FILE=, or ACCT= keyword specified (this match is implied to be a prefix match; ie: a TIP/30 userid of "FRED" is considered a match of JOB=FREDCOB)
  - OR the TIP/30 account number specified at logon matches the ACCT= keyword specified (exact match of all four characters of each).

The account number is the 4 character account number as given on the JOB statement of the job that created the spool subfile.

3.82.3. SPL Keywords

Following is a summary of the keywords that are recognized by the SPL program. Keywords provide information that is used to select the desired subfile entry.

Unless otherwise noted, keywords which involve matching operations (such as JOB= or PROG=) allow the use of TIP/30 standard prefix notation. For example, specifying JOB=*FRED means matching job names that begin with the string "FRED".

Some keywords provide information to the SPL program that changes the behaviour of the SPL program (for example: PAge=).

Upper case characters in the keyword are required characters; lower case characters are noise characters for readability — if such characters are provided they must appear in the proper location.

When multiple keywords are specified the SPL program treats the search as if all of the implied conditions are ANDed together. For example, specifying:

```
JOB=FRED STEP=3
```

means the job name must equal "FRED" and the STEP number must be 3 for the subfile to qualify for processing.

Account= Process subfiles with this job account number.

The account number is a positional parameter on the OS/3 // JOB statement or specified using a // OPTION ACN= statement.

The SPL program compares only the first 4 characters of Account number information.

Band= Process subfiles with this print band name.

The cartridge name is a parameter on the // LCB statement.

Cart= An alternative keyword for Band=.

The cartridge name is a parameter on the // LCB statement.

COlumn=n Specify the starting column number (relative to one).

Default is 1 — implying start at column 1.

CoPies=n Specify (for the Print command) the number of copies desired.

Default is 1.

DELete= Turn on/off automatic delete (for the **PRint** command).

Specifying DEL=YES causes the SPL program to delete a subfile after it is successfully printed (any error or interruption turns off the implied delete specification).

An alternative way to specify this is by using an option on the command line: SPL/DEL.

Default: DELETE=NO.

FIle= Process subfiles created with this LFD name.

This allows selection based on original LFD name.

FOrm= Process subfiles that specify this form name.

This keyword allows selection based on original form name.

Hold= Process subfiles in "Held" or "not-held" class.

Indicates the class of spool queue (held or not held). Specified as "Y" or "N".

Default: Hold=Y (examine subfiles that are HELD).

Job= Process subfiles with this job name.

JobNo= Process subfiles with this job number.

The summarize SPL command displays subfile job numbers that may be referenced by this keyword.

Label= Process subfiles created with this label.

This keyword allows selection based on // LBL name.

PAge=n Specify starting page number.

The summarize SPL command displays number of pages in the subfile. This keyword allows user to begin processing at a specific page number.

Prog= Process subfiles created by this program name.

Selection by EXEC name.

Remote= Process subfiles for this remote destination.

The value specified is a destination from the // DST statement or the // ROUTE statement.

STep= Process subfiles created by this step number.

The step number within a job.

USing=term

Route SPL output to alternate terminal.

SPL is started as an asynchronous process (via TIPFORK) on the specified terminal.

Device AUX1 is assumed.

To route printout to AUX2 of your terminal for example, specify the VIA= keyword instead.

The user should not explicitly specify his terminal name as the term parameter (SPL does not run in background).

This keyword is relevant only for the PRint command.

Via=term[,dvc]

Route printing to an auxiliary device on another terminal. The SPL program directs the output to the auxiliary device specified by sending the data to the specified terminal's aux device.

This allows the user to "use" some other terminal (temporarily) to perform the printing.

The difference between VIA= and USING= is that VIA= ties up your terminal (but allows you to interrupt the printing by pressing a function key).

The default dvc is AUX1.

This keyword is relevant only for the PRint command.

Via=dest Alternate use of the VIA keyword to specify that the output is to a print destination supported by TIPPRINT (see documentation of the TIPPRINT facility).

Examples of valid TIPPRINT destinations are: AUXn, d:xxxxxx, AUXnTTTT, ROLL, AUX0.

This keyword is relevant only for the PRint command.

3.82.4. SPL Program Operation

Since the first subfile that matches the specified criteria may not be the intended subfile, the SPL program always prompts the user to determine if the found subfile is to be processed.

When SPL finds the first subfile (of the class and queue specified) that matches the criteria specified by the keyword information, it displays all known information about that subfile. The subfile that is found may not be the intended one — especially if the keyword information was too vague.

SPL then prompts the user for confirmation that the subfile found is indeed the one wanted. If the user replies "Yes", the command is carried out; if the reply is "No", the search continues for the correct subfile.

While a subfile is listed at the user's terminal, the user may press [MSG WAIT] to interrupt the display. The user is then prompted with a continuation prompt.

In response to the continuation prompt, the user may tab to the appropriate choice and press transmit.

The user may change page number (forward or backward) and/or may change the starting column number. To do this, specify:

```
▸PAGE nnn [,ccc]
```

where nnn is the page number to proceed to and ccc is the new starting column number.

3.82.5. SPL Function Key Use

The SPL program recognizes the following use of function keys:

[MSG WAIT] Interrupt display on terminal.

A prompt is issued with a continuation query.

[F2] Re-display last command entered (can save some typing).

[F3] Re-execute last command entered (can save some typing).

3.82.6. DEL — Delete Spool Subfile

This command enables the user to select spool subfiles to be deleted.

Syntax:

```
DEL  [queue]  [,ALL]  [...keywords...]
```

Where:

queue	Optional positional parameter which specifies the desired spool queue (default is PR).
ALL	Optional positional parameter which indicates that ALL subfiles found that match keyword criteria are to be processed.
keywords	See "3.82.3. SPL Keywords" on page 3-254.

Example:

```
DEL ALL JOB=COB74
```

This example selects ALL subfiles with a job name "COB74" in the held class for possible deletion.

Additional Considerations:

The SPL program displays information about each subfile in turn and prompts the user for delete verification.

3.82.7. E — Terminate SPL Program

This command causes the SPL program to terminate normally.

Syntax:

```
E
```

Where:

No parameters required.

3.82.8. H — Display SPL Help

This command displays a summary of the SPL program command syntax.

Syntax:

```
H
```

Where:

No parameters required.

Error Conditions:

The help information may not be available or may have been deleted.

3.82.9. L — List Subfile on Terminal

This command lists (displays) selected spool subfiles on the terminal. Since print lines may be longer than the width of some terminals, the output from the list command is sensitive to the number of columns that the terminal is defined to support.

The SPL L command truncates the display to the number of columns the destination terminal is defined to support.

Syntax:

```
L  [queue]  [,ALL]  [...keywords...]
```

Where:

queue	Optional positional parameter which specifies the spool queue to be searched. Default is PR.
ALL	Optional positional parameter which indicates that ALL subfiles found to match are to be processed.
keywords	See "3.82.3. SPL Keywords" on page 3-254.

Example:

```
L  JOB=COB74 PROG=LNKEDT
```

This example selects for listing on the terminal any entry in the (held) PR queue that has a job name "COB74" and a program name equal to "LNKEDT".

3.82.10. LE — List Error Page

The LE command begins listing a spool subfile on the terminal starting with the page that contains the error diagnostics. This command looks in the correct location depending on the type of print file that it is processing: page 2 for COBOL-74 compilations, last page for assemblies, etc.

The command processing automatically searches for the correct job step for the specified job name or job number.

Syntax:

```
LE    [queue]   [,ALL]   [...keywords...]
```

Where:

queue	Optional positional parameter which specifies the spool queue to be searched. Default is PR.
ALL	Optional positional parameter which indicates that ALL subfiles found to match are to be processed.
keywords	See "3.82.3. SPL Keywords" on page 3-254. This command requires either the JOB name or JOB number be specified.

Example:

```
LE JOB=COB74
```

This example lists the error diagnostics for the job "COB74".

Additional Considerations:

This command performs (in effect) an "L" command starting at the "appropriate" page of the appropriate step of the job (see separate section describing the L command).

3.82.11. P — Print Subfile

This command prints selected spool subfiles to a designated printer destination.

The SPL program uses the VFB information that is stored in the operating system spool file to determine how to handle carriage control operations. The system VFB information is converted to the appropriate pseudo VFB information that is required by the TIPPRINT interface. For more information about how TIPPRINT handles pseudo VFB carraige control, see the documentation of the TIPPRINT subroutine in *TIP/30 Programming Reference — ARP-600-04*.

Syntax:

```
P  [queue]  [,ALL]  [...keywords...]
```

Where:

queue Optional positional parameter which specifies the spool queue to be searched. Default is PR.

ALL Optional positional parameter which indicates that ALL subfiles found to match are to be processed.

keywords See "3.82.3. SPL Keywords" on page 3-254.

Example:

```
P ALL JOB=COB74
```

This example selects for printing on the AUX1 printer subfiles in the (held) PR queue that have job name "COB74".

Example:

```
P JOB=TEST VIA=C:TEST
```

This example command (assuming that it is executed on a properly configured PC with a STEP/PEP interface) creates a file named C:TEST.PRN on the MS-DOS "C:" drive from the selected subfile.

The extension .PRN is automatically provided by the SPL program when it is used to create an MS-DOS file using the STEP/PEP file transfer facilities.

Additional Considerations:

The alternative spelling of the Print command "PTR" may be used to cause the print command to take note of the COL= keyword and to truncate the print lines at a maximum of 80 print positions.

3.82.12. PC — Print Subfile with Compression

The "PC" command prints selected spool subfiles to a designated printer destination and performs a specialized type of space compression technique.

Syntax:

```
PC  [queue]  [,ALL]  [...keywords...]
```

Where:

queue	Optional positional parameter which specifies the spool queue to be searched. Default is PR.
ALL	Optional positional parameter which indicates that ALL subfiles found to match are to be processed.
keywords	See "3.82.3. SPL Keywords" on page 3-254.

Example:

```
PC JOB=TEST VIA=C:TEST
```

This example command (assuming that it is executed on a properly configured PC with a STEP or PEP interface) creates a file named C:TEST.PRN on the MS-DOS "C:" drive from the selected subfile.

The extension .PRN is automatically provided by the SPL program when it is used to create an MS-DOS file using the STEP/PEP file transfer facilities.

Additional Considerations:

The space compression algorithm reduces multiple consecutive spaces with two bytes constructed as follows:

X'1D'	X'20' + count - 1

The first byte (X'1D') is a special marker that indicates that this is the first byte of a compression sequence.

The second byte is a binary value indicating the number of spaces removed. To compute the number of spaces that are to replace the two byte compression sequence, subtract X'20' from the value in this byte and add 1.

Example:

```
X'1D6F'
```

In the above example, the two bytes represent 80 removed spaces (because the difference between X'6F' and X'20' is X'4F', which is equal to 79 in base 10).

Note: *Two or more compression sequences can appear in series if the number of spaces removed is a large number.*

Supplied with the TIP/30 system is an MS-DOS-based program that you can use to "decompress" a print file created with the SPL program "PC" command. The name of the element in the TIP library that contains the PC spool output file decompression program is **TC$SPLO**. To create the PC program from that library element, execute the following command line from a PC that is connected to the TIP/30 system via a STEP or PEP board:

```
TIP?▶COPY,X  TIP/TC$SPLO,,C:D_PRESS.EXE
```

Of course, you can change the name of the PC program to any name you desire. If you execute the PC program without parameters, the following help information is displayed:

```
Usage:   d_press infile <outfile>


     - if  <outfile>  is not specified,
       the  <infile>  is overwritten

     - specify  <prn>  as the  <outfile>
       for output to the printer

     - specify  <con>  as the  <outfile>
       for output to the screen
```

3.82.13. PT — Print with Test Page

This command is a variant of the SPL "P" (print) command. The SPL program prompts the user to determine whether or not a test page is required. If the reply is "Yes", SPL prints a test page (similar to the test page generated by the batch output writer) on the specified auxiliary device. When the user (eventually) aligns the paper correctly and replies "No" to the "Test Page?" prompt, SPL then proceeds to print whatever subfiles were specified.

Syntax:

```
PT     [queue]   [,ALL]   [...keywords...]
```

Where:

queue Optional positional parameter which specifies the spool queue to be searched. Default is PR.

ALL Optional positional parameter which indicates that ALL subfiles found to match are to be processed.

keywords See "3.82.3. SPL Keywords" on page 3-254.

Example:

```
PT JOB=PAYROLL FORM=CHEX
```

This example prints the subfile that originated from the job "PAYROLL" with form name "CHEX". The user is prompted for test pages until he indicates the alignment is correct.

3.82.14. Q — End Program and Logoff

This command terminates the SPL program normally. If the SPL program was executing at program stack level one (ie: not called from another program) the user is logged off TIP/30.

Syntax:

```
Q
```

Where:

No parameters required.

3.82.15. R — Release Held Subfile

This command releases subfiles(s) that are currently on hold. This command is intended to be a mechanism to allow the user to release a held subfile that is now to be printed.

Syntax:

```
R  [queue]  [,ALL]  [...keywords...]
```

Where:

queue Optional positional parameter which specifies the spool queue to be searched. Default is PR.

ALL Optional positional parameter indicating that ALL subfiles that match are to be processed.

keywords See "3.82.3. SPL Keywords" on page 3-254.

Example:

```
R JOB=COB74
```

This example releases any subfile in the (held) PR queue that has a job name "COB74".

Additional Considerations:

To release all spool subfiles for a particular job, the **B*** transaction may be simpler to use.

3.82.16. S — Summarize SPOOL Queue Contents

This command lists (on the terminal) subfiles that exist in the specified class and queue. Candidate subfiles must match the selection keywords.

By using this command the user can browse through the spool file to determine which spool subfiles exist.

Syntax:

```
S   [queue]   [,ALL]   [...keywords...]
```

Where:

queue	Optional positional parameter which specifies the spool queue to be searched. Default is PR.
ALL	Optional positional parameter which indicates that ALL subfiles found to match are to be processed.
keywords	See "3.82.3. SPL Keywords" on page 3-254.

Example:

```
S  H=N
```

This example displays a summary of information about the subfiles that are not held (queued) in the PR queue.

Example of S command output:

```
SPL(1)?►S H=N J=TFM
Job=TFM STep=1 Prog=WRTSML FIle=PRNTR Account=A#TT FOrm=STAND1
       Band=63-STD JobNo=1985 PAge 0 of 2
Job=TFM STep=2 Prog=ASM FIle=PRNTR Account=A#TT FOrm=STAND1
       Band=63-STD JobNo=1985 PAge 0 of 17
Job=TFM STep=3 Prog=LNKEDT FIle=PRNTR Account=A#TT FOrm=STAND1
       Band=63-STD JobNo=1985 PAge 0 of 3
End of PRNTR queue HELD status.
SPL(1)?►
```

3.82.17. ST — Display SPOOL Status

This command displays the current state of the OS/3 Spool system.

Syntax:

```
ST
```

Where:

No parameters are required.

Example of ST command:

```
TIP?►spl
'TIP/30 Spool File Utility' - Version = 4.0 (89/09/01)
SPL(1)?►st
OS/3 Spooling Statistics
Testlines:Yes   Headers:No   Prtlog:Yes   Prtacct:Yes
    Burst:No   Compress:Yes  Update:Yes  Recovery:Closed
      Available space in spool file:25%
SPL(1)?►
```

3.82.18. W — Write Subfile to Edit Buffer

This command selects subfiles to be written to a TIP/30 edit buffer. The spool subfile data is copied to an edit buffer with the specified name.

Syntax:

```
W  [queue] [,buffer]  [...keywords...]
```

Where:

queue Optional positional parameter which specifies the spool queue to be searched. Default is PR.

buffer Optional positional parameter which names the output edit buffer. Default is "SPOOL".

The edit buffer is created with a group name equal to the user's first elective group specification.

keywords See "3.82.3. SPL Keywords" on page 3-254.

Example:

```
W  ,MYCOMP  JOB=COB74
```

This example creates an edit buffer named "MYCOMP" containing the contents of a (held) print subfile with job name "COB74".

Additional Considerations:

This command writes only 80 columns to the edit buffer. The COL= keyword may be used to some advantage.

3.82.19. WL — Write Subfile to Library Element

This command writes spool subfiles to a specified OS/3 library element.

Syntax:

```
WL  [queue] [,file/elt] [...keywords...]
```

Where:

queue Optional positional parameter which specifies the spool queue to be searched. Default is PR.

file/elt Optional positional parameters which specify the output library and element name.

Default is RUN/SPOOL.

keywords See "3.82.3. SPL Keywords" on page 3-254.

Example:

```
WL  ,TSTSRC/MYCOMP  JOB=COB74
```

This example writes to library TSTSRC, element MYCOMP, subfiles in the (held) PR queue that have job name "COB74".

Additional Considerations:

This command writes only 128 columns to the specified element. The COL= keyword may be used to some advantage.

3.82.20. WS — Write Summary to Library Element

This command writes spool summary information to the OS/3 library element RUN/SPOOL.

Syntax:

```
WS  [queue]  [...keywords...]
```

Where:

queue Optional positional parameter which specifies the spool queue to be accessed. Default is PR.

keywords See "3.82.3. SPL Keywords" on page 3-254.

Example:

```
WS   P=LIBS
```

This example writes a summary of spool file information to RUN/SPOOL. Only information concerning subfiles that specify Program=LIBS are written.

3.82.21. Invoking SPL from a Program

The SPL transaction may be invoked by a TIP/30 program to start SPL operations (usually printing!) on another terminal. TIP/30 provides standard program control subroutines to accomplish these tasks (see "Calling TIP/30 Utilities").

The procedures outlined in the above reference apply to the SPL program. There are some other considerations that are unique for the SPL program.

SPL expects the actual command (for example: PR JOB=*AR FORM=STAND1) to appear in the CDA-TEXT field of the CDA. The parameter fields of the CDA are more or less ignored by the SPL program with this exception:

CDA-PARAM (1) is checked to see if it contains exactly the string "SPL MUTE" (the characters "SPL" followed by a space, followed by the characters "MUTE"). If this is the case, SPL assumes that it is to refrain from sending any messages to the originating terminal. This specification prevents SPL from attempting to send "Printing completed" messages and the like.

Example of starting SPL at another terminal:

```
       MOVE SPACES       TO CDA.
       MOVE 'PR ALL JOB=*AR FORM=STAND1 VIA=AUX1'
                         TO CDA-TEXT.
       MOVE 'SPL MUTE'   TO CDA-PARAM (1).
****   MOVE PIB-UID      TO CDA-PARAM (2).
       MOVE 'T109'       TO PIB-TID.
       MOVE 'SPL'        TO PIB-TRID.
       CALL 'TIPFORK'.
       IF NOT PIB-GOOD ...
```

This example starts SPL on terminal T109. When SPL begins execution at that terminal, it attempts to print all subfiles with a job name that begins with "AR" and a form name of "STAND1" — since the program is running on T109, the specification VIA=AUX1 implies T109's auxiliary device 1.

The appearance of "SPL MUTE" (note the embedded space!) in CDA parameter 1 informs SPL that no informative messages are to be output by the SPL program.

The commented line illustrates moving a string to CDA parameter 2. SPL allows the calling program to insert a string in parameter 2 to represent the userid of the caller. This allows SPL to access subfiles that match the userid of the caller.

This subtle additional item may be needed because the userid (as it appears in the SPL program's PIB) is literally "*BYPtttt" when it has been invoked by TIPFORK — that userid (instead of the user's proper id) may prevent SPL from matching a subfile with the intended userid.

3.83. STARTUP — Startup Processing

The STARTUP transaction is intended to be a mechanism whereby the system administrator can schedule one or more transactions that are to be run when the TIP/30 system is initially started.

The STARTUP transaction may be specified as the system startup transaction by specifying STARTUP=STARTUP in the TIP/30 generation TIPGEN statement (or via the corresponding keyword in the TIP/30 job control stream).

The STARTUP program first attempts to open a DEFKEY file named: "STARTUP/FUNCTION/KEYS" — if there is no such DEFKEY file, the program attempts to open the library element: "TIP$LOD/STARTUP,S" (the TIP$LOD library was chosen because it is guaranteed to exist and the contents of that library are preserved across TIP/30 releases).

STARTUP "performs" each line of the input stream (whether it is a line of the DEFKEY file or the library element) as if the line was a standard TIP/30 command line.

Since the STARTUP program MUST run in background (and normally is executed at system startup time) any item that STARTUP performs must be capable of running in background.

The STARTUP transaction, and all transactions that are executed as a result of running that transaction, are executed with the pseudonym userid of "CONSOLE". If a user of that name is defined in the TIP/30 Catalogue, the security level and group memberships of that userid take effect. If the userid "CONSOLE" is not defined, the TIP/30 system assumes a security level of 1 (TECH) and GROUPS=ARC (membership in group TIPY is assumed as always).

For more information about defining a "CONSOLE" userid, see the section on TIP/30 Operator Console commands.

For example, assume that the library element TIP$LOD/STARTUP,S contained the following line:

```
LOADGDA
```

The STARTUP program would TIPSUB to the transaction named "LOADGDA" (which would also run in background) and would accomplish whatever it is intended to do (in this case, one presumes that the program loads some information into the Global Data Area).

A subtle point is that the STARTUP program opens the library element as a ".IN" file (see description of "Redirected input"). This means that any transactions that are "executed" by STARTUP may be followed by commands to that transaction. STARTUP will never "see" those lines because those lines are automatically "read" by the individual transaction.

The STARTUP program may also be given a transaction name of "SHUTDOWN" so that one could have both a STARTUP and a SHUTDOWN set of procedures. The names "SHUTDOWN" and "STARTUP" in the foregoing discussion are interchangeable.

Additional Considerations:

The STARTUP program displays a message on the operating system console which indicates when it is scheduling (or "performing") a particular subordinate transaction.

The TIP$LOD library is normally restricted to WRITE=NO in the TIP/30 catalogue. Before attempting to write to this library, the system administrator may have to adjust the entry in the TIP/30 catalogue to allow the editor to write to TIP$LOD.

If the STARTUP program cannot find the appropriately named DEFKEY file or the appropriate element in the TIP$LOD library, it will simply terminate gracefully.

3.84. STOP — Shutdown TIP/30 Immediately

This command causes TIP/30 to shut down immediately. It does not wait for all users to log off.

Syntax:

```
STOP
```

Where:

No parameters required.

Additional Considerations:

The system SHUTDOWN program (if specified) is not scheduled.

Under normal operating conditions, this command should only be issued after an "EOJ" command has been entered. "EOJ" is the preferred method of shutdown (see "3.31. EOJ — TIP/30 Shutdown" on page 3-90 and the console operator command "EOJ").

A "STOP" command may be necessary to force off users that are running programs that do not recognize that system shutdown was requested. See the description of the field PIB-SYSTEM in the documentation of the PIB (Process Information Block).

WARNING

Forcibly stopping the TIP/30 system may jeopardize transactions that are in progress. TIP/30 issues a console warning message if there are outstanding record locks when the system is halted by the STOP or CRASH directive. If transaction roll back is necessary, the operations staff must perform warm restart before using the files involved. For more information, refer to the discussion of recovery operations in the documentation of TIP/30 Generation, Maintenance and Installation.

3.85. SWTCH — Send Full Screen Message

The SWTCH program provides the capability of sending a message to one or more terminals or logged-on users in the network. The message to be sent is limited only by the screen size.

The SWTCH program is provided for compatibility with a similar facility provided by IMS.

The message will be sent to each destination as an unsolicited message — the recipient must press [MSG WAIT] to receive the message.

Syntax:

```
SWTCH  dest-list ; msg text
```

Where:

dest-list A list of destinations to send the message.

A destination may be either a terminal name or a userid or a prefix notation specification for either.

The word "ALL" is reserved to indicate all terminals.

There is no restriction that a destination must appear only once in the list - the message will be sent as many times as implied by the destination list.

A comma MUST be used to separate each destination from the next item in the list.

A semi colon (;) must be used to signal the end of the list.

Example: SWTCH mary,fred,T108; .....

msg text The text of the message to be sent.

The text begins immediately following the semi colon delimiter which marks the end of the destination list.

The text of the message may extend over the entire screen (if desired).

Example:

```
▶SWTCH T108,T109;



TIME FOR LUNCH

LETS GO FOR A PIZZA!
```

Additional Considerations:

The SWTCH program displays (on line 3 of the terminal) a confirmation message indicating the message was sent and showing any destinations which were not sent the message.

When constructing the text of the message it might be wise to deliberately avoid using line 3 of the screen. When SWTCH replies with the results, you could then repair the destination list (if necessary), clear line 3 and then send the message again to the revised destination list.

Error Conditions:

A destination is invalid if it is a user name and the user is not presently logged on, or if the terminal name is not found.

3.86. SYM — Schedule OS/3 Symbiont

SYM is a utility program which interfaces with the operating system: OS/3. It allows the user to submit requests to run symbionts and some other console commands in the same manner as the OS/3 console operator. Common commands include: RV (run a program), PR (start an output writer), HO (hold an OS/3 queue), etc.

An informational message is sent to the OS/3 operator console whenever a command is scheduled by SYM. The message informs the operator that a command was issued and also shows the user name and terminal name of the submitter.

The SYM program may be run interactively or may be given a single command on the command line. If SYM is run interactively the user is prompted for each command until an End or Quit command is given. If a command is provided on the command line, SYM attempts to execute that command and then terminates normally.

If the SYM program detects that it has been called via a transaction name other than "SYM" it assumes that the transaction name *is* the desired command and that the parameters on the command line are command line parameter information. This composite command is attempted and then SYM terminates normally.

Many alternate transaction names are provided in the TIP/30 Catalogue to invoke SYM using a specific transaction name as the command. Examples of this are: RV, BX, UNS, BE, and so on. For an example, see "3.74. RV — Run OS/3 Job" on page 3-239.

Syntax:

```
command  parameters
```

Where:

command The name of the desired symbiont or console command. OS/3 symbiont commands may be submitted (these include commands like: RV, BE, DE, and so on).

Refer to OS/3 console operator documentation for details concerning the use of these commands.

Commands that are not symbionts that SYM recognizes:

End End the SYM program.

F1...F22 End the SYM program.

GO	To issue a GO command for a paused OS/3 job.
PAuse	To pause an executing OS/3 job.
Quit	End the SYM program and logoff TIP/30.
SW	The operator SWitch command to switch executing job priority.
UNS	The operator UNS command to submit an unsolicited console key in to a job. See also "3.92. UNS — Unsolicited Console Keyin" on page 3-353.

parameters The appropriate parameters for the requested symbiont or console command.

Example of issuing a single command:

```
SYM PR BX,JOB=TIP30
```

This example starts a burst mode output writer to print any print spool files with a job name of "TIP30".

Example of using UNS:

```
SYM UNS M2,S DO,L,LIN3,NET1
```

This example issues an unsolicited command to ICAM (in this example, it is assumed to be named "M2") to down a line.

Example of using SW:

```
SYM SW MYJOB,+2
```

This example issues the console operator SW command to switch the executing priority of job "MYJOB" up two levels.

Example of using the SYM program interactively:

```
TIP?▶sym
Enter Command and parameters.
SYM(1)?▶be spl,job=ALLINSON
Enter Command and parameters.
SYM(1)?▶ca FOOEY,n
FOOEY    is not active. Can not CAncel.
Enter Command and parameters.
SYM(1)?▶rv testjob:j,,date=890101
Enter Command and parameters.
SYM(1)?▶e
TIP?▶
```

In this example, several SYM commands are issued before the interaction is terminated by the "End" command.

Additional Considerations:

The SYM program may be called from TIP/30 native mode programs by using a call to the TIPSUB subroutine. More information about doing this can be found in the documentation of the TIP/30 Program Control System — PCS.

When invoked in this manner SYM expects the command and parameters in free format in the text area of the CDA (bytes 73 through 152; field named CDA-TEXT).

If an error is detected, byte 73 of the CDA (the first byte of the field CDA-TEXT) is set to X'FF'; otherwise, byte 73 of the CDA is not altered. This facility is extremely useful for submitting OS/3 commands from an online program.

SYM allows the user to invoke the cancel symbiont (CA) but does not allow any attempt to cancel the currently executing TIP/30 job or any ICAM symbiont.

TIP/30 (as distributed) includes catalogue entries for a number of transactions that are quick ways of calling SYM to perform a single function. For example, there is a transaction named "RV" which references the SYM load module. The existence of this transaction means that the "RV" transaction can have a low enough security to enable programmers to use it, but that the more powerful SYM transaction could have a higher security level and thus be unavailable to programmers.

The use of individual symbionts may be restricted by using this technique.

3.87. SYS — Display OS/3 System Status

The SYS utility program displays the current status of the OS/3 environment.

Syntax:

```
SYS[/delay]  cmd
```

Where:

delay Command line option to set the refresh rate of the SYS program (in seconds).

Default: 20 seconds

Minimum: 3 seconds

cmd One of the following commands:

A	Similar to "J" (see below) except that symbionts and shared code modules are also listed.
End	End the SYS program.
J	List jobs currently running. For each job, the memory size, executing program, job step number, job number, elapsed CPU seconds and execution priority values are displayed. This command also indicates free memory regions.
Quit	End the SYS program and logoff TIP/30.
W	At "delay" second intervals execute the "J" function.
WA	At "delay" second intervals execute the "A" function.
W jobname	Repeat the "J" function (list jobs currently running) until the named job starts and subsequently terminates.

Example:

```
 SYS J           : display OS/3 job information
.SYS W COB74     : start background program to monitor
                   progress of job named "COB74".
```

Example output of SYS:

```
TIP?▶sys j
'System statistics' - Version = 4.0 (89/12/04) -  OS/3 13.00.S4
Job            Size   Exec     Step#    Job#     CPU        Priority Waiting
GUST            24K  ML$$GI      1      7303      0.1 Secs    4,H    Icam
TIPDEV       1,516K  TB$TIP      2      7308    594.5 Secs    1,H
TIPTST         992K  TB$TIP      2      7311    122.1 Secs    2,H
DMS            568K  DBMS        1      7358     11.5 Secs    6,N
TIP40          720K  TB$TIP      2      7364    218.5 Secs    1,P
TIP32B         732K  TB$TIP      1      7374     45.9 Secs    2,H
  Total free memory  1,776K   Largest region  1,768K

TIP?▶
```

Additional Considerations:

If SYS is run as a background program with the Wait function (see second example shown above), it notifies the initiating user with unsolicited messages when "jobname" has started and when "jobname" has terminated.

The unsolicited message that is sent to the terminal has the format shown in the following example:

```
TJ$COB74 running at 9:47:46   EXEC COBL74,   STEP#  2  ▶
```

(the word "running" is replaced by "terminated" in the message that is sent when the job terminates).

This allows the user to continue with other interactive activities while SYS monitors the batch job asynchronously in the background.

When SYS is running in continuous display mode, press MSG-WAIT to interrupt the display.

If SYS is entered with no command it defaults to the "J" display and interactively prompts for subsequent commands.

3.88. TCB — OS/3 Task Control Blocks

The TCB transaction is a utility program which displays task control blocks that are attached to the OS/3 operating system task switch list.

The program displays details about: job name, memory region in hex, size in hex, type, executing load module name, CPU time, account number, storage protect key, switch list and scheduling priority.

Priority numbers displayed are the actual displacement from the head of the switch list; hence the first user priority is 4. For transients and the supervisor overlay area (SOA), the number displayed in the account field is actually the transient, or SOA overlay ID. and the name in the program field is the overlay name.

Syntax:

```
TCB  [ W ]  [dest]  [wait]
```

Where:

W Optional parameter to cause the TCB program to periodically refresh the display on the screen until a function key or the MSG WAIT key is pressed. The rate at which the information is refreshed is controlled by the parameter following the "W".

dest This parameter is used to specify a standard TIPPRINT output destination for the output of the TCB program.

The defaults are:

- "AUX0" (full screen) for interactive users
- "ROLL" (line by line output to the terminal) if the "W" command is specified
- "PRNTR" for background users.

wait This parameter is used in conjunction with the "W" specification in parameter one. A numeric value from 5 to 60 (inclusive) may be specified. The value is taken as the number of seconds in the refresh interval.

Default value is 10 seconds (this value is also used if the value specified is not numeric or is not within the acceptable range).

Example:

```
TCB W
```

This command invokes the TCB program to display the current OS/3 TCB map. Because the firsp parameter is "W", the TCB program runs continuously and refreshes the screen display every 10 seconds until the program is terminated.

Example of TCB Display:

```
Continue?▶Yes  ▶No
                        .........OS/3 T.C.B. Map.........
  Name      Address      Size    Type      Program   Step   CPU     Acct Key  Pri
SY$STD00 006420-006493     0k Switcher                      .0          00  00
         01EAA8-01EB1B     0k Spooler                       .0          00  02
         006020-0064C7     1k S.O.A.      TO$LODPR          .0   832    00  00
AREA # 1 0064C8-006977     1k Transient   SV$RSLDI          .0    74    03  03
AREA # 2 006978-006E27     1k Transient   SV$RSLOD          .0   138    03  03
AREA # 3 006E28-0072D7     1k Transient   SV$RSLDI          .0    74    03  03
AREA # 4 0072D8-007787     1k Transient   SV$RECV0          .0   127    05  03
AREA # 5 007788-007C37     1k Transient   TO$SPFND          .0   949    03  03
AREA # 6 007C38-0080E7     1k Transient   SV$RSLOD          .0   138    03  03
AREA # 7 0080E8-008597     1k Transient   SV$RSLDI          .0    74    02  03
AREA # 8 008598-008A47     1k Transient   SV$RSLOD          .0   138    03  03
SL$$VT00 03DE00-03E8FF     2k Symbiont                      .0  (VR)    00  02
RC$$IS00 03E900-0476FF    35k Symbiont                     1.9  (IS)    00  03
SL$TCA00 04A200-04C9FF    10k Symbiont                      .0  (TW)    00  03
SL$DMS00 06EE00-0711FF     9k Symbiont                      .0  (DM)    00  08
SL@@CM00 07BA00-07E5FF    11k Symbiont                      .0  (CM)    00  04
ML$$C200 07E600-122AFF   657k Symbiont                      .0  (C2)    00  01
TIP32B   2F7000-3ADFFF   732k Batch Job  TB$TIP00    1     45.6  TIP3   07  05,H
TIP40    3AE000-461FFF   720k Batch Job  TB$TIP00    2    212.8  TIP    05  04,P
DMS      462000-4EFFFF   568k Batch Job  DBMS0000    1     10.8  DBMS   04  09,N
TIPTST   4F0000-5E7FFF   992k Batch Job  TB$TIP00    2    118.2  TIP3   03  05,H
```

Additional Considerations:

To discontinue the display with the wait parameter, press a function key or the MSG WAIT key.

3.89. TFD — Screen Format Definition

The TFD program provides an interactive facility which enables the user to define or modify a TIP/30 screen format. A TIP/30 screen format is a template used by online programs to control the display of information on a terminal.

The definition process is performed entirely online. The process is accomplished by interacting with the TFD program. The definition requires a number of steps (referred to as "passes"):

Pass 1	Select general format options.
Pass 1A	Select colour definitions (optional).
Pass 2	Compose the format (heading and data fields).
Pass 3	Verify location and size of non-heading fields.
Pass 4	Identify protected data fields (optional).
Pass 5	Override field attributes and/or supply default data.

The user may choose to have the TFD program display HELP information before each pass of the definition procedure. The HELP screens describe what the user must do in the upcoming definition pass.

Screen formats (or simply: formats) are referenced by an eight character format name. This format name must begin with an alphabetic character and cannot contain an imbedded space, slash or comma, but any other displayable graphic character may be used.

Formats are also defined according to the user group that may access the format. An online program simply requests the format by name; the correct group is selected by TIP/30 based on the user running the program. Thus, a program refers to a screen format by name, but the screen format actually supplied by MCS depends on the group membership of the user running the program.

For example, consider an order-entry program which may be used by either English-speaking or French-speaking users. Each user could be defined to be in either the ENGLISH or FRENCH user group. Two formats could then be created — one with English heading information and the other with French heading information.

The program would request the format by name and the TIP/30 system would select either the format in the ENGLISH group or the FRENCH group depending on the group membership of the user running the program.

An important point to realize is that the two screen formats just described must have the data fields defined identically since there is no way the program can determine which is being used!

TIP/30 formats contain two classes of information: heading and data fields. Heading information is usually informational in nature and is normally (but not necessarily) protected. Data fields contain variable information (either supplied by the program or entered by the terminal operator).

Unprotected data fields are usually used to enable the terminal user to enter data; protected data fields are usually used by the program to display information that is not intended to be changed by the user.

TIP/30 formats also provide a wide range of editing facilities on a field by field basis. A field may be defined to have certain attributes (numeric, upper case, etc). These attributes will be enforced by the TIP/30 Message Control System thus relieving the application program of the burden of extensive field checking.

Syntax:

① `TFD [new grp] [,new name] [,old grp] [,old name]`

② `TFU [old grp] [,old name]`

Where:

new grp The group name associated with the format being created.

Default: TIPY.

new name The name of the format which is to be created.

This parameter is required.

old grp The group name associated with an existing format which is either being updated or used as a base for defining a new format.

Default: TIPY.

old name The name of an existing format which is either being updated or used as a base for defining a new format.

Additional Considerations:

The following rules are applied in the interpretation of the command line parameters:

- If the transaction code is TFD and parameters three and four are omitted, assume the programmer wishes to define a new format from scratch.
- If all four parameters are supplied (to TFD) assume that the programmer wishes to create a new format, but use an existing format as a starting point.
- If the user attempts to create a format that already exists, TFD displays the error message "FORMAT CURRENTLY EXISTS - PRESS F4 TO UPDATE". The user may then press the F4 key to enter update mode.

- If no parameters are supplied and the transaction code was TFD, the following screen format will be displayed to simplify the entry of the required parameters:

```
TF$TFD0D   T I P / 3 0   S c r e e n   F o r m a t   D e f i n i t i o n   14:28

                                Group      Format
                                ========   ========
        New screen format name: ________   ________
          From existing format: ________   ________                        [_]
```

- If no parameters are supplied and the transaction code was TFU (TIP/30 Format Update), the following screen format is displayed to simplify the entry of the existing screen group and name:

```
TF$TFD0U    T I P / 3 0   S c r e e n   F o r m a t   U p d a t e          14:28

                                Group      Format
                                ========   ========
        Existing screen format: ________   ________                        [_]
```

Example of ways to invoke TFD:

```
① TFD
② TFD EDP,TEST
③ TFD EDP,TEST,TIP$Y$,TF$CAL
④ TFD ,,PAYROLL,PAY02001
⑤ TFU PAYROLL,PAY02001
```

Example ① causes TFD to display a screen format which allows the user to specify the type of definition process.

Example ② causes TFD to prepare to create a format named TEST for the group EDP.

Example ③ prepares to create a new format named TEST (for the group EDP) using the existing format named TF$CAL (from the group TIP$Y$) as a starting point.

Example ④ or ⑤ prepares to update an existing format named PAY02001 (for the group PAYROLL). Note that this may be accomplished either by specifying all the command line parameters for TFD, or by using the simplified clone transaction code TFU.

3.89.1. Display Intensities

Terminals normally support several display intensities. There usually is (at least) a normal display and a lower intensity display (sometimes this is called "alternate brightness"). Many terminals offer additional combinations of these intensities by allowing the user to select the intensity of both the foreground and background display.

The TIP/30 screen format definition procedure supports the following intensity specifications and attempts to achieve the desired results depending on the capabilities of the terminal that is using the screen format.

Light	"L" specifies light or low intensity. This is called "alternate brightness" on some terminals.
Normal	"N" specifies normal intensity.
Off	"O" specifies off intensity — the display will not be visible.
Reverse	"R" specifies reverse video (the characters are formed with 'dark' lines on a bright background).
Blinking	"B" specifies that the field is to blink — that is, alternate from high to low intensity.
Flashing	"F" specifies a combination of reverse video with the background alternating from BRIGHT to OFF intensity.
Grotesque	"G" specifies a combination of reverse video with the background alternating from BRIGHT to LOW intensity. This is a variant of Flashing.
Hideous	"H" specifies a combination of reverse video with the background alternating from LOW to OFF intensity. This is a variant of Flashing.
Shaded	"S" specifies a combination of reverse video with the background in LOW intensity.

3.89.2. Format Definition Options

The first pass of the definition procedure requires the user to select from a variety of definition options. The following screen format is displayed. If a new format is being defined, standard default values will be initially supplied. If the user is updating an existing format, the values placed in the screen format will be the values that were last used to define the format.

The user should fill in the required information and press XMIT to proceed to the next pass of the definition procedure.

Function key 1 (F1) may be used to redisplay this screen (in case the display was inadvertently altered). Pressing Message-wait (MSG WAIT) or Function key 2 (F2) will cancel the TFD process.

```
TF$TFD01   T I P / 3 0   S c r e e n   F o r m a t   D e f i n i t i o n   14:28

 Group: ________, Format: ________, Option character: _, SOE character: _

      The format is to be output starting at screen row __
            The cursor is to rest at location (row,col) __,__
     Partial update of format - only update screen rows __ through __
          Erase the screen before the format is output? _  (Y/N)
                    Set all user data fields unprotected? _  (Y/N)
   Automatic TAB STOP with each unprotected data field? _  (Y/N)
          HEADING intensity? _     DATA field intensity? _  (L)ight    (N)ormal
   Negative field intensity? _   ERROR field intensity? _  (R)everse  (B)link
                 Format to be used on a colour monitor? _  (Y/N)
       Set the CHANGE attribute ON for all data fields? _  (Y/N)
Set the RIGHT JUSTIFY attribute for all numeric fields? _  (Y/N)
  Override field attributes and/or supply default data? _  (Y/N)
  Are there uni-directional fields (ie. SFS emulation)? _  (Y/N)
     Enter the format identification character for RPG? _

  Generate COBOL copy book? _ (Y/N)  as library/element:  ________/________
          Print the format? _ (Y/N)           on printer:  ________

    Display HELP information between definition passes? _  (Y/N)       (_)
```

option character

This defines the character that is to be used by the user to identify data fields during pass 2 of the screen definition process (described in the next section).

This character is also used to represent "blink" characters in heading data (see following section describing heading definition codes).

Default: ^ (circumflex character).

SOE character

This defines the character which the user may use whenever a real SOE character is desired. Since the user cannot use a real SOE character in heading data, some character has to be sacrificed to represent an SOE.

Default: "\" (the backslash character).

start row

This field governs the starting row of the format. The user may specify a value between 1 and 24 inclusive. The format will automatically be transmitted starting at the specified row (when a program outputs the format).

Default: row 01.

cursor location

These fields specify the row and column number (respectively) where the cursor is to be placed when the program outputs the format.

By default, TFD will compute the row and column of the first character of the first unprotected data field.

Since the desired location of the cursor is often not known beforehand, the designer of a screen format usually lets this field default.

The MSGAR utility program has a command which allows the re-specification of the cursor location at any time after the screen format has been created.

partial update

These fields may be entered (only when updating a format) to indicate to TFD that you wish to only update a portion of the format. For example, entering 5,8 would inform TFU that you wished to update only lines 5 through 8 (inclusive) of the format.

Use of this facility can greatly reduce the amount of effort required to make minor changes to an existing screen format.

erase screen

Choose "Y" or "N" indicating whether the screen is to be erased before this format is output by a program.

Default: Yes.

The default is probably preferable unless the user is defining an upper portion of a split-screen application!

all unprotected

Choose "Y" or "N" indicating whether all of the data fields in the format are unprotected.

Default: Yes

If this field is entered as "N", the format definition process will be extended to include pass-4.

auto tab stop

Choose "Y" or "N" indicating whether all unprotected data fields are to have an automatic tab stop associated with the field.

Default: Yes.

If this option is specified "No", users of the format will not be able to use the TAB-FWD and TAB-BACK keys on the terminal to quickly move from data field to data field.

If this option is specified "Yes", data fields on an FCC-style terminal will have the TAB attribute set; on non-FCC terminals a TAB character will be automatically inserted in the first available space preceding the field.

The user may use hard tab characters (tab set) in heading data to force a tab in that position.

hdg intensity

This field indicates the desired intensity for heading data.

Choices are: L, N, R, B, O, F, S, G, or H (see "3.89.1. Display Intensities" on page 3-287).

Default: Light.

data intensity

This field indicates the desired intensity for data fields.

Choices are: L, N, R, B, O, F, S, G, or H (see "3.89.1. Display Intensities" on page 3-287).

Default: Normal.

negative intensity

This field indicates the desired intensity for numeric data fields when the field contents are a negative number.

Choices are: L, N, R, B, O, F, S, G, or H (see "3.89.1. Display Intensities" on page 3-287).

Default: Normal.

error intensity

This field indicates the desired intensity for error fields.

Choices are: L, N, R, B, O, F, S, G, or H (see "3.89.1. Display Intensities" on page 3-287).

Default: Blink.

colour monitor?

Choose "Y" or "N" to indicate whether this format is to be used on a colour display terminal (a personal computer or a UTS-60).

Default: No.

If this option is selected, the user will be required to select colour combinations corresponding to display intensities in Pass-1A (which will follow this pass).

change attribute

Choose "Y" or "N" to indicate whether the "changed" FCC attribute is to be automatically set for all data fields. This is normally not required unless the format is to be used on terminals that are set to TRANSMIT CHANged mode.

Default: No.

Applicable only to FCC terminals.

set right justify

Choose "Y" or "N" to indicate whether the "right justify" FCC attribute is to be automatically set for all numeric data fields.

Default: No.

Applicable only to FCC terminals.

field override?

Choose "Y" or "N" to indicate whether format definition Pass-5 is to take place.

Pass-5 allows the user to override the attribute(s) and/or supply default data for each individual data field.

Default: No.

uni-directional?

Does this screen format have any input-only or output-only fields?

Default: No

Input-only fields will NOT be sent data during a TIPMSGO (unless default data was provided for the field).

Output-only fields cannot be modified as a result of an input message.

RPG id character?

Does this screen format support an RPG identification character?

Specify the character (if any) to be used to identify this screen format to an RPG program.

RPG programs may wish to have the first byte of the MCS-DATA set to a specific "id" character to facilitate input decisions.

Copybook gen?

Choose "Y" or "N" to indicate whether the TFD program is to automatically generate a COBOL style COPY element which maps out the data fields of the

format.

Default: No.

The user may also specify the desired library and element name for the COPY book (assuming "Yes" was selected).

TFD will invoke the MSGAR program at the end of the definition procedure to accomplish this task.

Print format?

The user may enter "Y" or "N" to indicate whether the TFD program is to automatically print a hard copy listing of the format.

Default: No.

The user may specify the desired print destination. Default is the system printer (PRNTR); other choices include destinations such as "AUX1" or an MS-DOS file name — see the documentation of the TIPPRINT facility for a description of various TIPPRINT destination names.

TFD will invoke the MSGAR program at the end of the definition procedure to accomplish this task.

Help info?

The user may enter "Y" or "N" to indicate whether the TFD program is to display HELP information from this point on in the definition process.

Default: "Y" for novice users (those who have logged on TIP/30 fewer than 25 times), otherwise "N".

3.89.3. Format Colour Definition

This particular pass of the format definition process is optional and is only presented to the user if the COLOUR MONITOR option (in the previous pass) was specified as Yes.

The following screen format is displayed:

```
TF$TFD02   T I P / 3 0   S c r e e n   F o r m a t   D e f i n i t i o n   14:28


 The screen format can be defined to use up to 16 colour combinations.
 Select the background and foreground colours below. The first nine colour
 combinations correspond to monochrome intensities one through nine.
+--------------------------------------+----------------------------------+
|  Monochrome display:                 |  Colour display:                 |
|       Intensity  Field Definition Code |      Foreground on Background    |
|       =========  ==================== |      ========================    |
|    1  Shaded      S (reverse on low)  |      1 __ on __    2 __ on __    |
|    2  Off         O                   |      3 __ on __    4 __ on __    |
|    3  Normal      N                   |      5 __ on __    6 __ on __    |
|    4  Low         L                   |      7 __ on __    8 __ on __    |
|    5  Blinking    B                   |      9 __ on __   10 __ on __    |
|    6  Reverse     R (reverse on bight)|     11 __ on __   12 __ on __    |
|    7  Flashing    F (bright to off)   |     13 __ on __   14 __ on __    |
|    8  Flashing    G (bright to low)   |     15 __ on __   16 __ on __    |
|    9  Flashing    H (low to off)      |                                  |
|10-16  Not used in monochrome          |                              [_] |
+--------------------------------------+----------------------------------+
Standard colour codes are:
 BL-black  RE-red  GR-green  YE-yellow  BU-blue  MA-magenta  CY-cyan  WH-white
```

The user must select whatever colour combinations are desired to correspond with the "standard" intensity settings on the monochrome display.

For example, monochrome display number 4 is LOW intensity. The user could define colour combination 4 as WH/RE (which will be used whenever LOW intensity was called for). This specification means that any field that would normally appear as low intensity on a monochrome display would appear on a colour monitor with a WHite foreground on a REd background.

It is important to realize that the colour combinations that are defined here are the only colour combinations that may be used in the screen format.

Note: *If the screen format is to be used on Personal Computers with colour display adaptors, it is far more sensible to configure PEP or STEP (the Uniscope emulation PC software) to always select specific colours for various field attributes — see the appropriate documentation for the PC supplied CONFIG program.*

3.89.4. Format Composition

The second pass of the definition procedure requires the user to define the desired layout of the screen format on the terminal. If a new screen format is being defined (without using an existing format as a starting base) TFD will simply clear the screen and wait for the user to compose the format and press XMIT.

If a screen format is being defined with an existing format as a starting base, TFD will display the existing format and wait for the user to modify it and press XMIT.

To define the format, the user must enter heading information and "field definitions". Heading information is simply whatever text the user wishes to appear on the screen. Field definitions make use of "field definition codes" and "field editing codes".

There are also a number of special heading character strings which may be used to select standard system information to be displayed as heading data in the screen format. For example, a common desire is to have the current date appear as part of the title information of the format.

As will be shown, the user need not treat this as a data field (and suffer the burden of dealing with the current date as "another" data field); all that is required is to use the reserved string $MMDDYY$ (for example) in the desired location of the format. The TIP/30 output routines will automatically supply the current date (in the specified format) in that location whenever the format is output to a terminal.

What follows now is a description of the standard field definition codes, field editing codes and the special reserved heading strings which may be specified in the screen format.

3.89.5. Field Definition Codes

Data fields may be defined to the TFD program by using combinations of certain characters that are normally interpreted by TFD as field definitions. For example, a numeric field may be defined as ZZZZ9. This is intentionally similar to COBOL-74 formatted picture clauses.

Both numeric and alphanumeric fields may contain editing codes to specify certain automatic editing that is to be handled by the TIP/30 output routines. This editing is transparent to the program and the programmer.

Fields may not span a line of the screen; that is, the end of a physical line of the screen (80 characters) OR the appearance of some heading data signals the end of a data field.

Note that it is possible that a specific terminal may not support some of the features selected during the format definition. The TIP/30 output routines will accomplish whatever is possible with the terminal that is being used. The designer of the format may select options without much concern with the actual terminal hardware that will use the format.

When a field definition is entered during the TFD composition pass, the user may enter the codes as a continuous string of the appropriate length:

```
UUUUUUUUUU
```

or may elect to use an equivalent shorthand notation:

```
U(10)
```

The shorthand notation permits the specification of a length within parentheses after the definition code character.

The following restrictions apply to this feature:

- the replication factor can be applied only to homogeneous fields; — mixing field definition codes is not permitted
- the data field starts at the initial field definition code position
- there must not be any other characters within the implied range of the replication factor (eg: if you specify X(30), there must be 25 spaces following the five character string: "X(30)").
- the characters included within the parentheses must be numeric and represent a value from 2 to 132 inclusive. Leading zeroes are permitted.

If all these conditions are not met, TFD treats the code as heading information.

A...A Define upper case alphabetic field.

On both input and output, any alphabetic character will be automatically translated to the equivalent upper case character.

On FCC terminals, the ALPHABETIC FCC will be set. This will prevent any nonalphabetic character from being input.

NO software enforcement of alphabetic-only is provided on non-FCC terminals.

B...B Define upper case field — "blinking".

Treated the same as a "U" field, except that the field will be displayed as a blinking field. Note that a field which is all spaces cannot blink!

E...E Define error field.

Error fields are (by definition) output-only areas that are typically used to display error or informational messages. These are not real data fields.

F...F Define upper case field — "flashing".

Treated the same as a "U" field, except that the field will be displayed in REVERSE video on a background that alternates from BRIGHT to OFF intensity.

G...G Define upper case field — "grotesque".

Treated the same as a "U" field, except that the field will be displayed in REVERSE video on a background that alternates from BRIGHT to LOW intensity.

H...H Define upper case field — "hideous".

Treated the same as a "U" field, except that the field will be displayed in REVERSE video on a background that alternates from LOW to OFF intensity.

L...L Define upper case field — "low intensity".

Treated the same as a "U" field, except that the field will be displayed in LOW intensity.

N...N Define upper case field — "normal intensity".

Treated the same as a "U" field, except that the field will be displayed in NORMAL intensity.

O...O Define upper case field — "off".

Treated the same as a "U" field, except that the field will have the FCC attribute "display off".

R...R — Define upper case field — "reverse video".

Treated the same as a "U" field, except that the field will be displayed in REVERSE video.

S...S — Define upper case field — "shaded".

Treated the same as a "U" field, except that the field will be displayed as REVERSE video on a LOW intensity background.

U...U — Define upper case field.

On both input and output, any alphabetic character will be automatically translated to the equivalent upper case character. The field may contain any displayable graphic character.

The field will be displayed in whatever intensity was specified (in Pass-1) for DATA fields.

X...X — Define alphanumeric field.

Any printable characters may be input or output in this format. No translation or checking is performed.

Z...Z — Define numeric digit with zero suppression.

A "Z" field definition code represents a digit of a numeric field which will be displayed as a space on output if the digit is a leading zero. The digit will be forced to a valid digit on input (space becomes a zero).

9...9 — Define numeric digit.

A "9" field definition code represents a digit of a numeric field which will be unconditionally displayed on output. The digit will be forced to a valid digit on input (space becomes a zero).

2...2 — Define numeric digit — blank if not input.

A "2" field definition code represents a digit of a numeric field which will be unconditionally displayed on output. The digit will be input as a space if it was not entered from the terminal.

If a "2" definition is used in a field, the entire field will be returned as spaces if it is not input from the terminal. This allows the combination ZZZ,ZZ2.22 to be edited as ZZZ,ZZ9.99 with the additional consideration that the field will be blank if not input.

Example:

```
UUUUUUUUUUUUUUU   - 15 character upper case field
EEEEE             -  5 character error field
UUUXXX            - two (adjacent) alphanumeric fields
                    each is 3 characters long.
ZZZZ9             - 5 digit numeric field with leading zero
                    suppression (on output). Ie: 34 will
                    display as ___34 (underscores illustrate
                    leading spaces).
```

3.89.6. Field Editing Codes

Numeric data fields may be defined to include various types of editing. The editing that occurs is transparent to the application program that uses the screen format. The edit characters are supplied by the TIP/30 MCS output routines and are removed on input.

comma insertion

Commas may be placed in the numeric field to specify comma insertion.

Example: 99,999

The character used in the format definition depends on a TIP/30 generation option; If DECIMAL=COMMA was specified in the TIP/30 generation, the comma and decimal point roles are reversed.

decimal point

A decimal point may be placed in a numeric field to indicate decimal point alignment.

The decimal point is used on input to align the field in the program MCS area.

Example: ZZ9.99

The character used in the format definition depends on a TIP/30 generation option; If DECIMAL=COMMA was specified in the TIP/30 generation, the comma and decimal point roles are reversed.

leading minus

A leading minus sign may be specified to cause a (floating) leading minus sign to appear for negative numeric values.

Example: -ZZ,ZZ9

trailing minus

A trailing minus sign may be specified to cause negative numeric values to appear with a minus sign after the last digit.

Example: ZZ,ZZ9-

parentheses

Parentheses may be placed around a numeric field to cause negative values to be surrounded by a floating left parenthesis and a fixed right parenthesis.

Example: (ZZ,ZZ9)

CR symbol

A numeric field may be suffixed by the two characters "CR" (upper or lower case) to cause the symbol "CR" to appear as a suffix to a negative value.

Example: ZZ,ZZ9CR

DB symbol

A numeric field may be suffixed by the two characters "DB" (upper or lower case) to cause the symbol "DB" to appear as a suffix to a negative value.

Example: Z,ZZZ,ZZ9DB

/ insertion

A numeric field may be edited with imbedded slash characters ("/"). This is often specified for date fields.

Example: 99/99/99

: insertion

A numeric field may be edited with imbedded colon characters (":"). This is often specified for time fields.

Example: 99:99:99

floating currency

A floating currency symbol may be specified for a numeric field. The currency symbol (as specified in the TIP/30 generation) will be floated in front of the first digit displayed.

Example: $ZZ,ZZ9.99

The character used while composing the format definition is the dollar sign character. When the screen format is used, the run time MCS routines use the character that is specified in the TIP/30 generation parameter CURRENCY=.

Table 3-14. Examples of MCS Editing

Format	Value	Display
ZZ,999	00127	___127
ZZ,999.99	0001280	___012.80
$Z,ZZ9.99	000108	____$1.08
99/99/99	120480	12/04/80
ZZ/99/99	080480	8/04/80
(ZZ,ZZ9)	0122M	(1,224)
ZZ,ZZ9CR	0122M	1,224CR
ZZ,ZZ9CR	01234	1,234

Note: *A negative value is represented internally in memory as a zoned numeric value with an "11 over punch" of the right most digit. (Eg: the digits 0 through 9, as the right most digit of a negative value, would be stored as the characters "}JKLMNOPQR" — X'D0' through X'D9').*

3.89.7. Heading Definition Codes

Heading information in a screen format is normally entered exactly as desired at format definition time. Some variable information is often useful to display and is best handled as heading data (rather than output-only data fields): current date/time, format name, user-id, etc.

This section documents a number of reserved symbols that represent data that is to be supplied by the TIP/30 MCS output routines.

The character strings are replaced by the appropriate data when the screen format is output to the terminal. This reduces the amount of effort required in the application program.

_ An underscore character can be used to designate a "cursor resting location". This character will (by default) be set to an unprotected underscore that is NOT a data field.

Example: [_]

Note: *This use of the underscore does nothing more than leave an unprotected area in the screen format as a resting place for the cursor. The "resting place" is not a field and therefore does not enter into any calculations at run-time about how much data is transmitted from the terminal and the subsequent indications that are given to the program.*

Refer to the description of the TIPMSGI subroutine call in the MCS section of this documentation for additional information about the input data count.

In any case, a preferable way to designate a cursor resting location may be the definition of a final (dummy) one character field at the end of the screen format — the programs that use the screen format would, of course, ignore whatever data appears in that field.

option character

The option character defined in Pass-1 may be utilized in heading information to represent a "blink" character. The option character will be replaced by either a left or right blink character (X'1C' or X'1D').

This option character defaults to the circumflex character (^).

TFD normally starts by using the left blink character and alternates with a right blink. In the case where there are adjacent blink characters, no alternation is done (two or more adjacent blinks will be the same type of blink character).

Example: ^This is important^
results in: «This is important»

SOE character

The character defined in Pass-1 as a substitute for an SOE may be used as a heading character to imbed SOE characters within a screen format.

Example: Next account number \UUUUUUUU

$MMDDYY$

This string of characters will be replaced by the current date in the format: MM/DD/YY (8 characters).

Example: Date: $MMDDYY$ == Date: 4/17/84

$DDMMMYY$

This string of characters will be replaced by the current date in the format: DD MMM YY (9 characters) where the MMM field will be an abbreviation of the month in English (ie: JAN, FEB, MAR ... DEC).

Example: Date: $DDMMMYY$ == Date: 17 APR 84

$YYMMDD$

This string of characters will be replaced by the current date in the format: YY/MM/DD (8 characters).

Example: Date: $YYMMDD$ == Date: 84/04/17

$HHMM

This string of characters will be replaced by the current time in the format: HH:MM (5 characters).

Example: Time: $HHMM == Time: 09:30

$USERID$

This string of characters will be replaced by the 8-character user-id of the user that is logged on the terminal that is using the screen format.

Example: $USERID$ == ALLINSON

$TID

This string of characters will be replaced by the name of the terminal where the screen format is being displayed (4 characters).

Example: $TID == T107

$FRMTID$

This string of characters will be replaced by the name of the screen format (8 characters).

Example: $FRMTID$ == PAY02001

$TRANID$

This string of characters will be replaced by the transaction code (8 characters) of the program currently displaying the screen format.

Example: $TRANID$ == PAYUPD

3.89.8. Secure Information

There is often a need for a program to display certain information ONLY if the user has a particular security clearance. For example, a payroll inquiry program may wish to display the salary information only if the user has a TIP/30 user security higher than (say) 65.

To facilitate this, the TFD program allows the designer of a screen format to specify that an area of the screen is not to be displayed unless the user has a specific security.

The following reserved character strings may be specified in the format definition to control the display of the screen format according to the user's security.

An important point is this: once a security specification is encountered by TFD, it applies to ALL following heading and data information until another security specification is encountered.

$<nnn This character string is a dollar sign followed by a less-than symbol followed by a 1, 2, or 3 digit number representing the security level.

The heading and data information from this point on in the screen format will not be displayed unless the user's security level is numerically less than or equal to the number specified.

Example: $<65 will allow the display of following headings and data only if the user has a security of 1 through 65.

$=nnn This character string is a dollar sign followed by an equal symbol followed by a 1, 2, or 3 digit number representing the security level.

The heading and data information from this point on in the screen format will not be displayed unless the user's security level is exactly equal to the value specified.

Example: $=1 will allow the display of following headings and data only if the user has a security of 1.

Note that specifying $=0 effectively hides the following information from all users of the screen format.

This technique may be used to hide fields which are not yet implemented by the program (but someday will be filled in with data).

$> This character string is simply a dollar sign followed by a greater-than symbol.

This string is used to "turn off" the security mechanism or to "reveal" all of the following headings and data.

Additional Considerations:

These security specifications may be placed throughout the screen format. Note that the screen positions that are occupied by these security specification strings are turned to spaces (and are, therefore, "wasted").

The heading and data fields that are "concealed" by the action of a security specification string are NOT "squeezed" out of the format — the screen space they would occupy is filled with (protected) spaces.

Once a screen format is defined with security specification(s), it cannot be modified by a user unless that user is able to "see" all fields in the screen format. For example, a user with security level 15 could not modify a screen format if that format contained security specification(s) which were numerically lower than 15.

Any attempt to update a screen format that violates the above rule would result in TFD/TFU terminating with a "Security violation" message.

An exception to the above rule is areas that are defined as $=0 or $<0 — such areas are not subject to security checking since no one can actually have a security level of zero!

Example of Secure Field Definition:

```
              PAYROLL INFORMATION

               NAME:     UUUUUUUUUUUUUUUUUUUU
              TITLE:     UUUUUUUUUU
$<50        SALARY:     $ZZZ,ZZ9.99       $>
          HOLIDAYS:           ZZZ9 days
```

In this example format, a user who has a TIP/30 security level that is numerically less than or equal to 50 will "see" the heading and data associated with the SALARY; a user with lower security would see protected spaces where the SALARY information line could have been.

Note the use of $> to discontinue the concealing of information.

If a screen format was defined in this manner, a user would have to have a security level numerically less than or equal to 50 to be able to update the screen format.

This ensures that a user that has access to the TFD program cannot update an existing screen format to circumvent the security provisions already established by another (higher security) user!

3.89.9. TFD Line Copy

TFD supports a type of "cut and paste" operation available during screen definition. Unlike traditional cut and paste operations, the text that is initially identified is not deleted from the screen — it is merely "marked". This section describes the mechanics of this facility.

Note: This facility is operational only during the initial screen definition pass of the format definition utility (pass 2).

The MARK operation involves the following steps:

1. Place the cursor anywhere in the first line that is to be "marked".
2. Press the [F3] key and observe that a start-of-entry character appears in the last column of the line above the line where the cursor was placed.
3. Move the cursor to the end of the text that you wish to "mark" and press the [XMIT] key (the SOE character that was output by step 2 is removed from the screen).

The screen text that was just marked may be recalled (as many times as desired) by performing the following PASTE operation.

The PASTE operation causes previously marked text to be output on the screen. To recall the marked text, perform the following steps:

1. Move the cursor to the desired start location of the text.
2. Press the [F4] key and observe that the text is output at the point where the cursor was located and the cursor location has been advanced to the end of the recalled text.

The PASTE operation overlays any data that may already be on the screen but only for the number of lines that were marked.

The example that follows illustrates the use of this procedure to copy a set of lines of screen definition.

Assume that the screen data shown below has been manually entered on the screen during the initial TFD format definition step:

```
                          TEST TFD MULTI-LINE COPY


   Name: U(25)
Address: U(30)
         U(30)
```

The intent is to make several copies of the Name and Address lines (including the blank separator line which follows the fields). Place the cursor anywhere on the line which contains "Name: U(25)" and press the [F3] key.

The result is the following display:

```
                          TEST TFD MULTI-LINE COPY

   Name: U(25)
Address: U(30)
         U(30)
```

Position the cursor in the blank line which follows the second address field and press the XMT key.

The text from the cursor location back to the SOE character is marked by TFD and may now be recalled as many times as needed. The SOE character is removed from the screen.

Position the cursor after the blank line which follows the second address field and press F4 to recall the marked lines.

```
                          TEST TFD MULTI-LINE COPY

   Name: U(25)
Address: U(30)
         U(30)

   Name: U(25)
Address: U(30)
         U(30)
```

The cursor may be repositioned and F4 pressed to retrieve the marked text any number of times. The mark and paste operation may be repeated to copy different sets of lines.

3.89.10. Identify Heading Data

The third pass of the definition procedure requires the user to identify what areas of the screen format are indeed data fields. The TFD program has been presented with the composition of the format — the heading information and the field definition codes and possibly editing codes. The user and the TFD program must now agree on the size and location of the data fields, edit information, error fields, special heading data and security specifications.

This step is necessary because the TFD program may not be able to distinguish some heading information from (what the user intended to be) data fields.

At the start of this pass, TFD will redisplay the composed screen format. TFD will place the option character (as specified in Pass-1 — usually a circumflex) in each position of what it considers to be a data field, error field, editing character, special heading field or security specification.

Note: *It is the responsibility of the person defining the screen format to make sure that any incorrect guesses are repaired.*

TFD often guesses correctly, but there are situations where a string of characters in the format (as composed in Pass-2) will appear to be part of a data field when it is in fact simply data (or the opposite situation).

The user should examine the entire screen and be certain that the option character (usually a circumflex) appears only in those areas which are not simple heading data.

If any errors are observed, the user can simply insert an option character or remove it as the situation warrants (it is not necessary to replace an erroneous occurrence of the option character with exactly what should be there — TFD merely uses the presence of an option character in a particular location to indicate that the location is part of a data (or error) field.

After the screen has been corrected (if necessary), the user should press [XMIT] to continue the definition process.

Pressing [MSG WAIT] cancels the definition process.

3.89.11. Identify Unprotected

The fourth pass of the definition procedure is optional. It appears only if the user had specified (during Pass-1) that all data fields are not unprotected. If all data fields are not unprotected then the user must now indicate to the TFD program which areas of the screen are to be protected and which areas are to be unprotected.

Although this pass materializes as a result of a question about data fields, the user may now effectively override the protection characteristics of the entire screen. For example, it can be convenient to "unprotect" various heading areas.

The TFD program displays the following screen format:

```
TF$TFD03   T I P / 3 0   S c r e e n   F o r m a t   D e f i n i t i o n   14:28

  ▶Place ^ over the UNPROTECTED areas of the format.

        Are most of the data fields unprotected? Y (Y/N)   [_]
```

Enter a "Y" if most of the data fields in the screen format are to be unprotected; otherwise, an answer of "N" is more appropriate.

TFD now expects you to place the option character (usually a circumflex) in the areas of the screen which are to be unprotected. If you replied "Y" to the above question, TFD will return your screen format (as composed) with the option character already in place of all data fields. If you answered "N" to the above question, TFD will return your screen format (as composed) without any option characters.

You must now ensure that the option character appears in all areas of the screen format that are to be unprotected. As was the case in Pass-3, it is not necessary to replace an undesirable occurrence of the option character by the exact character that should occupy that location — the absence of the option character is sufficient information for the TFD program.

A technique that is often employed by application programs is the inclusion of unprotected (heading) information in a screen format (usually with a leading SOE character). When this (heading) information is explicitly unprotected, the user of the screen format can simply place the cursor at the end of the area and press the XMIT key.

This is a simple way to imbed "commands" in a screen format that is normally used just to display data.

3.89.12. Override Field Attributes

The fifth pass of the definition procedure is optional. It appears only if the user had specified (during Pass-1) that he wished to override field attributes. This pass will step through the format field by field, allowing the user to accept the current attributes of the individual field or to make alterations to the field's attributes.

Each field has a number of attributes that are associated with it as a result of the screen format definition up to this point:

- display intensity / colour / blinking (mutually exclusive!)
- protected / unprotected
- automatic tabbing?
- return blanks if not entered (numeric fields only!)?
- automatically set changed FCC attribute?

A field has a specific display intensity (Normal, Low, etc). This intensity is translated into a specific colour combination if the format is being used on a colour monitor and the format was explicitly declared to be for a colour monitor (Pass-1A). The user may alter the intensity/colour combination in this pass.

A field may be forced to have (or not have) an automatic TAB stop.

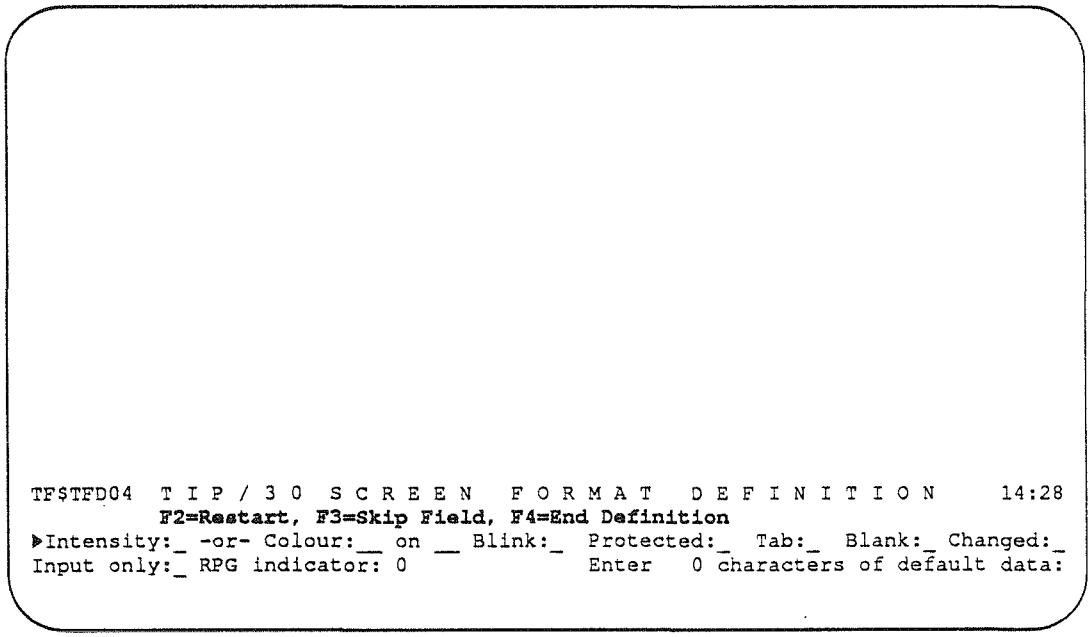
A numeric field can be forced to return spaces to the user program (if it is not entered on input). For example, a field defined as ZZZZZ can be forced to return spaces when not entered (even though it was not defined with the "2" field definition code).

The user can supply default data for the field (this is the only way this default data may be specified). Default data is supplied by the TIP/30 output routines whenever the field is not explicitly output by the program (ie: the field is past the length of data implied by MCS-COUNT on output).

WARNING

When a screen format is updated, the default data is not retained for lines that are being updated. When updating a format that has default data, the user is advised to make use of the partial update facility to minimize the amount of default data that must be entered again.

TFD outputs the following screen format at the start of pass-5:



The upper portion of the screen format contain the first 19 lines of the screen format that is being defined; the last 5 lines are temporarily used by pass-5 to solicit the attribute changes.

The user must designate the changes to be made to the attributes for the field which is currently being processed. The current field is highlighted in the upper portion by the appearance of blink markers on either side of the field.

The user may alter the attributes as desired and press XMIT to make the alterations and proceed to the next field.

The available function keys are advertised by a message in the screen format:

F1 Refresh the screen.

F2 Restart the definition process at the beginning of Pass-2 (This is not a good key to press by accident!)

F3 Accept this field's attributes and move to the next field.

F4 Terminate Pass-5 processing.

When the user needs to examine or alter the attributes of a field that is on or after line 20 of the screen, TFD will "invert" the display to show the bottom 19 lines of the user's format and temporarily use the top 5 lines to solicit attribute changes:

```
TF$TFD05  T I P / 3 0  S C R E E N  F O R M A T  D E F I N I T I O N    14:28
          F2=Restart, F3=Skip Field, F4=End Definition
▶Intensity:_ -or- Colour:__ on __ Blink:_  Protected:_  Tab:_  Blank:_ Changed:_
Input only:_ RPG indicator  0              Enter   0 characters of default data:
```

The various attribute changes are now described in detail.

intensity The intensity field will be unprotected by the TFD program if the format being defined is not declared to be for a colour monitor. The user may change the desired intensity of the field.

Choices are: L, N, R, B, O, F, S, G, or H (see "3.89.1. Display Intensities" on page 3-287).

This field and the (following) colour field are mutually exclusive attributes — only one of the choices will be available.

Only the field intensity attribute may be changed here. A field cannot be changed from type U to type X or from type U to numeric (for example).

colour The colour field(s) will be unprotected by the TFD program if the format being defined was declared to be for a colour monitor. The user can alter the selection of the background:foreground colour combination.

Note that the user may only specify a combination that was already specified during Pass-1A. You may not "create" a new colour combination at this stage.

Refer to the description of Pass-1A for the available colours and their mnemonics.

blink The user may specify Y to force this field to blink.

This attribute change only applies to colour fields.

protected The user may specify Y to force this field to be a protected field.

Tab The user may specify Y to cause the FCC for this field to be built with the automatic TAB flag set on.

If Y is specified, the FCC for this field will include the ability to TAB to this field.

blank This attribute applies only to numeric fields!

The user may specify Y to indicate that the TIP/30 input routines are to put spaces (rather than zeroes) in the MCS-DATA area for this field if the field is not entered by the terminal operator.

Changed The user may specify Y to indicate that the FCC for this field is to be built with the "changed" flag set on.

If Y is specified, a terminal that has the XMIT option set to CHAN will consider this field changed (even if the terminal operator does not alter it) and transmit all changed fields up to and including this field.

input only The user may specify Y to indicate that this field is an input-only field.

This attribute is only applicable to SFS (Screen Format Services) emulation.

default data

The user is given the opportunity to enter up to the specified number of characters of default data for the field.

If no data is entered, TFD will assume that there is no default data for this field.

A special notation is allowed to enable the user to indicate that the field is to be filled with a particular character — enter the option character as the first character of default data and follow the option character with whatever character is to be used as a fill character.

Eg: entering ^? means fill with question marks

3.89.13. Screen Format Summary

At the end of the format definition, TFD (or TFU) will display the screen format shown below to provide summary information about the format that was just created or updated (the phrase "created" will change to "updated" if an existing screen was updated).

```
TF$TFD06   T I P / 3 0   S c r e e n   F o r m a t   D e f i n i t i o n
           ============  ============  ============  ====================

    Group: ________  Format: ________    created on xx/xx/xx at xx:xx:xx

    The format contains ______ data field characters
                    and ______ error field characters.

    The message format is ______ bytes. (Re: MSGPOOL)

    The cursor resting location is at row __, column __.
```

The group and name of the screen format will be shown along with the date and time that the format was created (or updated).

The number of bytes of DATA characters is shown. This value determines the sum of the lengths of all the DATA fields in the screen format and, in effect, defines the length of the data area needed by an application program that will use the format.

The number of bytes of ERROR field characters is also shown. This represents the sum of the lengths of all ERROR fields that are in the screen format. This information is required by the application programmer who wishes to construct and send ERROR (or informational) messages when this format is in use at a terminal.

The number of bytes that the format occupies internally is shown. This number is important to the system programmer who must decide on the size of buffers to be used for TIP/30 screen format pooling (TIP/30 generation parameter MSGPOOL=).

The cursor resting location is identified as a row and column location. This will either be the first character of the first unprotected data field (by default) or will be set to the value selected at definition time.

The number of DATA bytes, number of ERROR bytes, and the format size are also displayed by the MSGAR online program and the batch program which sorts and lists the TIP/30 catalogue and MCS file.

3.89.14. Format Definition Example

To illustrate the definition process, assume that we wish to create a screen format that is to appear as follows:

```
$FRMTID$        TIP/30  FORMAT  DEFINITION   EXAMPLE  SCREEN  FORMAT     $DDMMMYY$
                    EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE


        Customer Name:  UUUUUUUUUUUUUUUUUUUUUUUUUUUUUU

             Address:  UUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
                       UUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
                       UUUUUUUUUUUUUUUUUUUUUUUUUUUUUU

          Postal Code:  U9U 9U9

            Area code:  (999)
            Telephone:  999-9999

      Current Balance:  ($ZZZ,ZZ9.99)
                                                      <_>
```

This screen format is to be named "EXAMPLE" and is to be defined to be in the group "TIPY". Some of the more important facts about the screen format are:

- The screen is to display the screen format name in the upper left corner and the current date in the upper right hand corner
- The screen is to be used only for inquiry activity; all data fields are output-only and, therefore, are to be protected
- The Customer name and address fields are upper case fields 30 characters long
- The postal code is standard Canada Post format: letter, digit, letter, a space, digit, letter, digit
- The area code is to be displayed with parentheses around it (this is NOT parentheses around a negative number!)
- The telephone number is to have a "hard" minus sign between the exchange and set number (nnn-nnnn)
- The current balance field is to have a floating currency symbol, zero suppression and surrounding parentheses if the value is negative.

In order to begin the definition of such a screen, we must invoke the TIP/30 Format Definition program (TFD):

```
TIP?▶TFD
```

TFD determines that we did not supply any parameters on the command line and prompts with the following screen format:

```
TF$TFD0D   T I P / 3 0   S c r e e n   F o r m a t   D e f i n i t i o n   14:28

                                Group      Format
                                ========   ========
          New screen format name: TIP$Y$__  ________
            From existing format: TIP$Y$__  ________                        [_]
```

We wish to create a new screen format and we do not have an existing format to use as a model, so we fill in the screen as shown below and press the XMIT key:

```
TF$TFD0D   T I P / 3 0   S c r e e n   F o r m a t   D e f i n i t i o n   14:28

                                Group      Format
                                ========   ========
          New screen format name: TIP$Y$__  EXAMPLE_
            From existing format: TIP$Y$__  ________                        [_]
```

TFD now responds with the screen format for Pass-1 (Select general format options):

```
TF$TFD01   T I P / 3 0   S c r e e n   F o r m a t   D e f i n i t i o n   14:28

  Group: TIP$Y$__,  Format: EXAMPLE_,  Option character: #,  SOE character: \

       The format is to be output starting at screen row _1
             The cursor is to rest at location (row,col) __,__
      Partial update of format - only update screen rows _1 through 24
           Erase the screen before the format is output? Y  (Y/N)
                    Set all user data fields unprotected? N  (Y/N)
    Automatic TAB STOP with each unprotected data field? Y  (Y/N)
          HEADING intensity? L    DATA field intensity? N  (L)ight   (N)ormal
    Negative field intensity? N   ERROR field intensity? B  (R)everse (B)link
                  Format to be used on a colour monitor? N  (Y/N)
        Set the CHANGE attribute ON for all data fields? N  (Y/N)
 Set the RIGHT JUSTIFY attribute for all numeric fields? N  (Y/N)
   Override field attributes and/or supply default data? N  (Y/N)
   Are there uni-directional fields (ie. SFS emulation)? N  (Y/N)
      Enter the format identification character for RPG? _

   Generate COBOL copy book?  N (Y/N)  as library/element:  RUN_____/EXAMPLE_
          Print the format?  N (Y/N)           on printer:  PRNTR___

     Display HELP information between definition passes? N  (Y/N)       (_)
```

TFD has recognized the group and format name that we specified earlier and that most of the default values are correct.

However, ALL our data fields are to be protected, so we must alter the value specified as the answer to the question "Set all user data fields unprotected?" to be "N" rather than the default "Y".

For the purpose of this example we have also changed the option character from a circumflex to the character # (as shown on line 3 of the format above).

After these changes are made, we press the XMIT key.

TFD now clears the screen (since we did not specify an existing screen format to use as a starting base) and waits for us to compose our format. We enter the following into the blank screen provided:

```
$FRMTID$        TIP/30  FORMAT  DEFINITION   EXAMPLE  SCREEN  FORMAT     $DDMMMYY$
                    EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE

        Customer Name:  UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU

              Address:  UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
                        UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
                        UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU

          Postal Code:  U9U 9U9

            Area code:  (999)
            Telephone:  999-9999

      Current Balance:  ($ZZZ,ZZ9.99)
                                                        <_>
```

We have used the heading definition codes $FRMTID$ and $DDMMMYY$ to signify that we want the format name and the current date to appear in the indicated locations on the screen.

We have also defined the Error field (EEE...EE) where we would like it and used various field definition and editing codes to define the various data fields.

Even though this screen format is intended to be used just to display data, we have provided a cursor resting area — more for cosmetic appearances than any other reason.

We now place the cursor immediately to the right of the last non-blank character of our format (in this case, right after the cursor resting location we have defined) and press XMIT to complete Pass-2 (composition pass).

TFD now re-displays the composed screen format with the option character (the #) in place of all characters that TFD thinks are either part of an Error field, data field, special heading code, security code or editing code:

```
########       TIP/30  FORMAT  DEFINITION   EXAMPLE  SCREEN  FORMAT     #########
                  ########################################

        Customer Name:  ##############################

              Address:  ##############################
                        ##############################
                        ##############################

          Postal Code:  ### ###

            Area code:  #####
            Telephone:  ###-####

      Current Balance:  #############
                                                      <_>
```

TFD has correctly over struck (with the #) all fields **except** the area code field (TFD thinks that the field is to have parentheses around it if its negative — we wanted the parentheses to be "hard" heading characters).

We must now correct TFD's impressions! We move the cursor to the area code field and change the first and last # back into left and right parentheses (we need not use an actual parenthesis, but it is visually convenient!).

```
########       TIP/30  FORMAT  DEFINITION   EXAMPLE  SCREEN  FORMAT     #########
                ########################################

        Customer Name:  ##############################

              Address:  ##############################
                        ##############################
                        ##############################

          Postal Code:  ### ###

            Area code:  (###)
            Telephone:  ###-####

      Current Balance:  #############
                                                  <#>
```

and press the XMIT key.

TFD now displays the following screen format (this format actually is defined to start at line 19 of the CRT — this temporarily overlays the bottom 5 lines of our format):

```
TF$TFD03   T I P / 3 0   S c r e e n   F o r m a t   D e f i n i t i o n   14:28

  ▶Place # over the UNPROTECTED areas of the format.

       Are most of the data fields unprotected? Y (Y/N)   [_]
```

Since ALL of our data fields are to be protected, and we are being asked to place the # option character over those fields which are to be unprotected, we now change the (default) response from "Y" to "N" and press the XMIT key.

TFD re-displays the screen format as it was originally composed (without any of the # characters) since we replied "No" to the previous question. (Note that the defined cursor resting location is returned with the # option character since it is assumed to be unprotected space in the screen format).

```
$FRMTID$        TIP/30  FORMAT  DEFINITION   EXAMPLE  SCREEN  FORMAT     $DDMMMYY$
                    EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE


        Customer Name:  UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU

             Address:  UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
                       UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
                       UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU

          Postal Code:  U9U 9U9

            Area code:  (999)
            Telephone:  999-9999

      Current Balance:  ($ZZZ,ZZ9.99)
                                                      <_>
```

We do not have any modifications to make; no fields are to be unprotected so we can simply press the XMIT key.

The format definition process is now complete. The following summary screen is displayed:

```
TF$TFD06   T I P / 3 0   S c r e e n   F o r m a t   D e f i n i t i o n
           ===========   ===========   ===========   ==================

    Group: TIP$Y$    Format: EXAMPLE     created on 06/20/84 at 13:11:31

    The format contains     144  data field characters
                    and      40 error field characters.

    The message format is    338 bytes. (Re: MSGPOOL)

    The cursor resting location is at row 17, column 54.
```

From following information is shown on the summary screen:

- There are indeed 40 error field characters
- The total size of the data fields (exclusive of any editing characters) is 144 bytes
- The format is stored internally in 338 bytes of storage. This value may be of interest to the system programmer when he is determining an optimum selection for the MSGPOOL= TIP/30 generation option
- The cursor resting location was computed as row 17, column 54.

The cursor resting location defaulted to the defined cursor resting location since there are NO other unprotected fields on the screen.

One could, of course, elect to alter this cursor resting location by using the MSGAR program:

```
TIP?▶MSGAR CUR EXAMPLE 24 80
```

A TIP/30 program that uses this screen format might display data in the following manner:

```
EXAMPLE        TIP/30  FORMAT  DEFINITION   EXAMPLE  SCREEN  FORMAT    10 JAN 89


        Customer Name:  Allinson-Ross Corporation

              Address:  4250 Sherwoodtowne Blvd
                        Mississauga, Ontario
                        CANADA

          Postal Code:  L4Z 2G6

            Area code:  (416)
            Telephone:  848-2030

      Current Balance:     $1,000.00
                                                  <_>
```

3.90. TLIB — Librarian Services

TLIB is a utility transaction program that provides online librarian facilities. TLIB manipulates OS/3 library elements, TIP/30 edit buffers, terminal auxiliary devices (cassette, diskette, printer), and MS-DOS files on Personal Computers (PC).

To manipulate MS-DOS files, TLIB must be executed at a Personal Computer that is logged on TIP/30 and is using the Computer Logics Personal Emulator Package (PEP) software and hardware (or equivalent). The user is responsible for verifying that the PC has a compatible UNISCOPE interface.

TLIB is able to manipulate library elements that are source (S), macro (M), proc (P), object (O) or load (L).

For certain commands TLIB recognizes three pseudo library element types: directory ("D"), fast directory ("F") and Edit Buffer ("E"). Directory type "D" implies the library header information including module name, module type, comments, date and time stamp (similar to a LIBS table of contents listing). Directory type "F" implies only the module name and module type. Type "E" (Edit Buffer) indicates that the specification applies to a TIP/30 edit buffer.

TLIB may be used interactively or may be given a single command on the command line. If a single command is given on the command line TLIB will attempt only that command and terminate. When used interactively, TLIB prompts the user for each command.

Syntax:

① `TLIB[/options]`

② `TLIB[/options] cmd parameters`

Syntax ① is used to start TLIB in interactive mode. TLIB prompts the terminal operator for each command until an "end" or "quit" command is given.

Syntax ② is used to execute a single TLIB command.

3.90.1. TLIB Commands

If TLIB detects that it has been called with a transaction name other than "TLIB", it assumes that the transaction code is the command and does not treat the first parameter as a command (the implied command is inserted and the parameters are internally shifted right one position).

TLIB recognizes the following commands:

Table 3-15. TLIB Commands

Command	Description
F1	In interactive mode, recall last command for cannibalization and possible re-submission.
BACK/RECOVER*	Reactivate the previous version of an element.
COPY*	Copy an element or edit buffer to an element, auxiliary device or MS-DOS file.
DELETE/ERASE*	Delete a library element.
DIR*	Detailed directory of a library.
END	End TLIB program.
FDIR*	Fast directory of a library.
HELP	Display help information on terminal.
JOB	Submit an element or edit buffer to the OS/3 input reader.
LIST*	List (on the terminal) an element or edit buffer.
PRINT*	Print a listing of an element or edit buffer.
PUNCH	Punch an element or edit buffer.
QUIT	End TLIB program and logoff.
SETOF	Set TLIB option OFF.
SETON	Set TLIB option ON.

Note: *In the above table, commands that are suffixed with an asterisk are also implemented as TIP/30 transactions (with the transaction name equal to the TLIB command name). The definition of separate transaction "clone" names permits the system administrator to restrict access to certain commands to users with appropriate security levels and also permits the use of commands (like COPY) as stand alone transactions.*

3.90.2. TLIB Options

TLIB recognizes the following options (whether supplied on the command line or referenced by a SETON or SETOF command in interactive mode). The initial state of these options is "not in effect". Command line options are processed by TLIB from left to right.

Table 3-16. TLIB Options

Option	Description
A	ASSEMBLER mode — use columns 1-72.
C	COBOL mode — use columns 1-72.
H	Turn on PCXFER routine's Hexification.
I	Include module comments (as first record) of MS-DOS output.
K	COPY: prompt for output element comment.
L	Print with line numbers.
M	Do not print heading lines.
N	Do not print title page.
O	COPY: Omit overwrite prompt. DELETE/ERASE: Ignore edit buffer changed flag.
Q	Do not display any messages (Quiet mode).
R	RPG mode use columns 1-74; columns 1-5 set to spaces.
S	Scratch input edit buffer.
X	Turn on PC's hexification.
Y	Use PCXFER file compression (when copying data to an MS-DOS PC file.
1...8	PC display number for PC data transfer.

Note: *Hexification (options "H" and "X") is the process of converting data bytes that are not valid graphic characters into two graphic characters that represent the hexadecimal portions of the character. For example, X'B3' becomes the two graphic characters "B" and "3". Of course, the hexification can occur in the opposite direction too — two bytes become one hexadecimal byte value.*

Additional Considerations:

Some options are mutually exclusive (for example, the language specification options like 'R" and "C").

3.90.3. TLIB Input and Output Specifications

The following tables illustrate the various formats that the parameters to TLIB can take. P1 refers to the first parameter after the TLIB command. P2 is the next parameter and so on.

Table 3-17. TLIB Input Specifications

Parameters P1, P2, P3	Description
omitted	If parameters 1 through 3 are omitted, the terminal is used as an input device. Input is solicited line by line until (MSG WAIT) is pressed to signal end of input.
File/elt [,type]	Input OS/3 Library element. Default is source type "S". Other choices are Macro, Proc, Load module, Object module, Directory, Fast directory, Name of Proc, Internal Symbol Dictionary of Load module.
group/name,E	TIP/30 edit buffer specification. Pseudo type code "E" indicates a TIP/30 Edit Buffer.
d:ffffffff.eee	MS-DOS file specification: **d:** MS-DOS drive identifier **ffffffff** MS-DOS file name **.eee** MS-DOS file extension
AUXn,,,	Auxiliary device specification "n" is device number (1 through F) Some AUX devices are capable of write operations only and cannot be specified as an input device.
UNIX:	Input comes from a UNIX system.

Table 3-18. TLIB Output Specifications

Parameters P4, P5, P6, P7	Description
omitted	If no output specifications are provided, TLIB output is directed to AUX0 (full screen display via TIPPRINT).
File [,element] [,type]	OS/3 library element specification Default element name is input element name (P2) Default type is input type (P3)
Group [,name], E	TIP/30 edit buffer specification Group: the edit buffer group name: the edit buffer name (required if group is given) type (must be "E" for Edit buffer)
d:ffffffff.eee	MS-DOS file specification: **d:** MS-DOS drive identifier **ffffffff** MS-DOS file name **.eee** MS-DOS file extension
AUXn	Auxiliary device specification "n" is auxiliary device number (1 through F)
printername	A printer name acceptable to TIPPRINT. See description of TIPPRINT in the documentation of the TIP/30 File Control System — FCS.
queue [,lbl1] [,lbl2] [,H or R]	OS/3 spool file specification **queue** OS/3 spool queue (RDR, RDR96, RDR128, RBPIN) **lbl1** First part of the LBL name (max 8 characters) **lbl2** Second part of the LBL name (max 8 characters). lbl1 and lbl2 are concatenated to form the LBL name. **R** Create the spool queue entry with RETAIN. **H** Create the spool queue entry with HOLD.

3.90.4. COPY — COPY Data

The TLIB Copy command is able to copy data to and from a wide range of locations. Not only can standard OS/3 library elements be copied, TIP/30 edit buffers, auxiliary devices, MS-DOS files and the OS/3 spool file can be utilized.

The COPY command recognizes the following input specifications:

- OS/3 library element
- OS/3 library directory
- MS-DOS file on a PC
- terminal auxiliary device (cassette, UTS diskette)
- TIP/30 edit buffer
- UNIX file
- the terminal (entered at keyboard).

The COPY command recognizes the following output destinations:

- OS/3 library element
- MS-DOS file on a PC
- terminal auxiliary device (cassette, UTS diskette, printer)
- TIP/30 edit buffer
- TIPPRINT printer destination
- UNIX file
- OS/3 RDR Spool sub file.

The combinations of input and output specifications make the syntax of the COPY command rather cumbersome to illustrate. However, a general rule is: the first 3 parameters pertain to the input specification, parameters 4 through 7 pertain to the output specification.

Syntax:

```
Copy  P1 [,P2] [,P3] ,P4 [,P5] [,P6] [,P7]
```

Specify the input specification for COPY (parameters P1, P2 and P3) from the following table:

Parameters P1, P2, P3	Description
omitted	If parameters 1 through 3 are omitted, the terminal is used as an input device. Input is solicited line by line until MSG WAIT is pressed to signal end of input.
File/elt [,type]	Input OS/3 Library element Default is source type "S".
group/name,E	TIP/30 edit buffer specification Pseudo type code "E" indicates TIP/30 Edit Buffer.
d:ffffffff.eee	MS-DOS file specification: **d:** MS-DOS drive identifier **ffffffff** MS-DOS file name **.eee** MS-DOS file extension
AUXn,,,	Auxiliary device specification "n" is device number (1 through F) Some AUX devices are capable of write operations only and cannot be specified as an input device.

Specify the output specification for COPY (parameters P4, P5, P6 and P7) from the following table:

Parameters P4, P5, P6, P7	Description
omitted	If no output specifications are provided, TLIB output is directed to AUX0 (full screen display via TIPPRINT).
File [,element] [,type]	OS/3 library element specification Default element name is input element name (P2). Default type is input type (P3)
Group [,name], E	TIP/30 edit buffer specification Group: the edit buffer group name: the edit buffer name (required if group is given) Pseudo type code must be "E" for Edit buffer.
d:ffffffff.eee	MS-DOS file specification: **d:** MS-DOS drive identifier **ffffffff** MS-DOS file name **.eee** MS-DOS file extension
AUXn	Auxiliary device specification "n" is auxiliary device number (1 through F)
printername	A printer name acceptable to TIPPRINT. See description of TIPPRINT in the documentation of the TIP/30 File Control System — FCS.
queue [,lbl1] [,lbl2] [,hold]	OS/3 spool file specification **queue** OS/3 spool queue (RDR, RDR96, RDR128, RBPIN) **lbl1** First part of the LBL name (max 8 characters) **lbl2** S Second part of the LBL name (max 8 characters). lbl1 and lbl2 are concatenated to form the LBL name. **hold=R** Create the spool queue entry with RETAIN. **hold=H** Create the spool queue entry with HOLD.

Note: *When a reader queue element is created by COPY, it may be accessed using standard OS/3 batch utilities (such as DATA) by using job control specifying:*

For RDR or RDR96	For RDR128:
// DVC 30	// DVC 130,I
	// VOL X(NOV) or RDR128
// LBL xxxxxxxxyyyyyyyy	// LBL xxxxxxxxyyyyyyyy
// LFD zzzzzzzz	// LFD zzzzzzzz

xxxxxxxxyyyyyyyy

Represents the label name (or names) specified when the file was created.

zzzzzzzz

The appropriate LFD name for the input file for the batch program.

If the reader queue entry is created in the HOLD state, it is necessary to begin the spool queue entry and give the waiting job a GO command:

```
BE SPL,RDR,FILE=xxxxxxxxyyyyyyyy
GO jobname
```

Example of COPY commands:

Command	Description
COPY JCS/TIP30,,SYSGEN	Copies the element TIP30 (default type "S") from library JCS to library SYSGEN as element name TIP30.
COPY JCS/TIP30,,SYSGEN/OLDTIP	Copies the element TIP30 (default type "S") from library JCS to library SYSGEN as element name OLDTIP.
COPY PG11,,E,AUX1	Copies the edit buffer PG11 to the terminal AUX1 device (presumably a printer).
COPY JCS/TIP30,,B:/TEST/PRN	Copies library element TIP30 from library JCS to MS-DOS file B:TEST.PRN
COPY TEST,,E,RDR,jobxyz,cards	Copies edit buffer TEST to the OS/3 reader as LBL name "JOBXYZCARDS"
COPY SRC/PAY040,S,C:PAY040.COB	Copy source element PAY040 from library SRC to a PC file named C:PAY040.COB

Additional Considerations:

The following options affect the COPY command:

A	ASSEMBLER mode — use columns 1-72.
C	COBOL mode — use columns 1-72.
H	Turn on PCXFER routine's Hexification.
I	Include module comments in first record of MS-DOS output.
K	Prompt for comment text if output is a library element.
O	If output library element already exists, overwrite it without issuing a confirmation prompt.
Q	Do not display any messages (Quiet).
R	RPG mode — use columns 1-74; columns 1-5 set to spaces.
S	Scratch input edit buffer.
X	Turn on PC's hexification.
1...8	Specify PC display number for PC data transfer. Default is no alteration of PC control page.

Error Conditions:

The input file, element or edit buffer may not be found or the output file may not be available for use.

3.90.5. DELETE — Delete Library Element

This command is used to delete an element from an OS/3 library or to delete a TIP/30 edit buffer (also see the discussion included with TLIB's RECOVER command in "3.90.15. RECOVER — Activate Previous Version" on page 3-348).

Syntax:

① `DELete  P1 [,P2] [,P3]`

② `ERASE  P1 [,P2] [,P3]`

Where:

Specify the item to be deleted from the following table:

Parameters P1, P2, P3	Description
File/elt [,type]	Input OS/3 Library element Default is source type "S".
group/name,E	TIP/30 edit buffer specification Pseudo type code "E" indicates TIP/30 Edit Buffer.

Example:

```
DELETE  JCS/MYJOB
```

Deletes element "MYJOB" from library "JCS".

Additional Considerations:

ERASE may be used as a synonym for DELete.

The following options affect the DELETE command:

O — If input is an edit buffer, ignore changed flag (do not prompt for overwrite confirmation).

Q — Do not display any messages (Quiet)

Error Conditions:

The specified element may not exist or the file cannot be accessed.

3.90.6. DIR — Display Library Directory

This command displays a directory of an OS/3 library (or some subset of an OS/3 library) at the terminal. A line, containing the module's name, type, comment and date and time last changed, will be displayed for each module selected.

Syntax:

```
Dir  file [,prefix] [,printer]
```

Where:

file	The selected library name as defined in the TIP/30 catalogue.
prefix	An element name prefix to be used to select some subset of the elements in the library. If the prefix parameter is omitted, the default is assumed to be "*" — all elements.
printer	The destination printer (default is AUX0 — full screen display). Other possibilities are, for example, PRNTR, AUX1 or AUX1*BYP etc.

Example:

```
DIR  TIP,*TH$
```

Displays a directory of all elements with names that begin with "TH$", in the library file catalogued with the name "TIP".

Example of DIR command output:

```
Continue?▶Yes  ▶No
 Listing: TIP/*TH$,D  89/02/23  23:00  DIRECTORY
TH$TQLMO,S    VER-002.D HELP FOR 'TQLMON'       84/01/13  14:41
TH$TQLC,S     VER-004.D HELP FOR 'TQL C'        84/01/13  16:11
TH$TQLCO,S    VER-002.D HELP FOR 'TQL CO'       84/01/13  16:16
TH$TQLCP,S    VER-003.D HELP FOR 'TQL CP'       84/01/13  16:18
TH$TQLWP,S    VER-003.D HELP FOR 'TQL WP'       84/01/16  11:29
TH$TQLW,S     VER-003.D HELP FOR 'TQL W'        84/01/16  11:30
TH$TQLUP,S    VER-003.D HELP FOR 'TQL UP'       84/01/16  11:30
TH$TQLUF,S    VER-003.D HELP FOR 'TQL UF'       84/01/16  11:31
TH$TQLUC,S    VER-003.D HELP FOR 'TQL UC'       84/01/16  11:31
TH$TQLU,S     VER-003.D HELP FOR 'TQL U'        84/01/16  11:32
TH$TQLSP,S    VER-003.D HELP FOR 'TQL SP'       84/01/16  11:32
TH$TQLS,S     VER-003.D HELP FOR 'TQL S'        84/01/16  11:33
TH$TQLQP,S    VER-005.D HELP FOR 'TQL QP'       84/01/16  11:34
TH$TQLQ,S     VER-003.D HELP FOR 'TQL Q'        84/01/16  11:34
TH$TQLPP,S    VER-003.D HELP FOR 'TQL PP'       84/01/16  11:35
TH$TQLP,S     VER-003.D HELP FOR 'TQL P'        84/01/16  11:35
TH$TQLNP,S    VER-003.D HELP FOR 'TQL NP'       84/01/16  11:36
TH$TQLNF,S    VER-003.D HELP FOR 'TQL NF'       84/01/16  11:36
TH$TQLN,S     VER-003.D HELP FOR 'TQL N'        84/01/16  11:37
TH$TQLM,S     VER-003.D HELP FOR 'TQL M'        84/01/16  11:38
TH$TQLLP,S    VER-003.D HELP FOR 'TQL LP'       84/01/16  11:39
TH$TQLL,S     VER-003.D HELP FOR 'TQL L'        84/01/16  11:39
```

3.90.7. END — End TLIB Interaction

The End command terminates the TLIB program in a normal fashion.

Syntax:

```
End
```

Where:

No parameters required.

3.90.8. FDIR — Display Abbreviated Library Directory

This command displays a "fast" directory of an OS/3 library (or some subset of an OS/3 library) at the terminal. Up to six element names are listed on each output line; the module name and type is displayed for each module selected.

Syntax:

```
Fdir  file [,prefix] [,printer]
```

Where:

file The selected library name as defined in the TIP/30 catalogue.

prefix An element name prefix to be used to select some subset of the elements in the library. Default is list all elements.

Elements are listed without regard to the type of the element.

printer The destination printer (default is AUX0 — full screen display). Other possibilities are, for example, PRNTR, AUX1 or AUX1*BYP etc.

Example:

```
FDIR  TIP,*TH$
```

Displays a directory of all elements with names that begin with "TH$", in the library file catalogued with the logical file name "TIP".

Example of FDIR command output:

```
Continue?▶Yes  ▶No
 Listing: TIP/*TH$,F  89/09/11  15:51  DIRECTORY
TH$TQLMO,S   TH$TQLC,S    TH$TQLCO,S   TH$TQLCP,S   TH$TQLWP,S   TH$TQLW,S
TH$TQLUP,S   TH$TQLUF,S   TH$TQLUC,S   TH$TQLU,S    TH$TQLSP,S   TH$TQLS,S
TH$TQLQP,S   TH$TQLQ,S    TH$TQLPP,S   TH$TQLP,S    TH$TQLNP,S   TH$TQLNF,S
TH$TQLN,S    TH$TQLM,S    TH$TQLLP,S   TH$TQLL,S    TH$TQLDP,S   TH$TQLD,S
TH$QED,S     TH$APTAD,S   TH$APTCU,S   TH$APTLD,S   TH$APTLM,S   TH$APTMT,S
TH$APTSM,S   TH$APTMD,S   TH$DLL,S     TH$APTDE,S   TH$MAS,S     TH$LOGON,S
TH$MSG,S     TH$TFD,S     TH$CALEN,S   TH$MAILL,S   TH$SWTCH,S   TH$WMI,S
TH$SKEL,S    TH$$$$$$,S   TH$TQL,S     TH$USERS,S   TH$IDA,S     TH$LOOK4,S
TH$SPL,S     TH$SORT,S    TH$SET,S     TH$WARNG,S   TH$JCL,S     TH$PURGE,S
TH$PMDA,S    TH$NEWUS,S   TH$GO,S      TH$JBQ,S     TH$ACCES,S   TH$ALLOC,S
TH$APB,S     TH$APTPU,S   TH$BANNE,S   TH$BASIC,S   TH$BCP,S     TH$CC,S
TH$CMS,S     TH$CPAGE,S   TH$SYMXX,S   TH$SYM,S     TH$EOJ,S     TH$DBD,S
TH$CONNE,S   TH$DCF,S     TH$DEBUG,S   TH$DEFKE,S   TH$DOTIN,S   TH$DISAB,S
TH$DIE,S     TH$DOC,S     TH$DOF,S     TH$EDTRS,S   TH$UP,S      TH$LOGOF,S
TH$CCA,S     TH$FREE,S    TH$FCLOS,S   TH$FSEPM,S   TH$FSE01,S   TH$FSE03,S
TH$FSE04,S   TH$FSE05,S   TH$FSE06,S   TH$GROUP,S   TH$HARDW,S   TH$HELP,S
TH$HELPE,S   TH$MEM,S     TH$MENU,S    TH$MENUA,S   TH$MENUD,S   TH$HANGU,S
TH$MODE,S    TH$MSDOS,S   TH$NET,S     TH$MSGSH,S   TH$NEWPA,S   TH$NOTE,S
TH$QCLEA,S   TH$RDR,S     TH$RPG,S     TH$SCR,S     TH$DDPCN,S   TH$SFSCN,S
TH$SHUTD,S   TH$STATU,S   TH$SUBMI,S   TH$SYS,S     TH$TIPFL,S   TH$RELOA,S
TH$TCB,S     TH$TSP,S     TH$WHOSO,S   TH$TLIB,S    TH$APT,S     TH$FSESU,S
TH$FSEFK,S   TH$FSE02,S   TH$FSE,S     TH$DD,S      TH$TIPPR,S   TH$AFT,S
```

3.90.9. HELP — Help for TLIB Commands

The Help command invokes the TIP/30 HELP system to display syntax help for the TLIB program and it's clone transactions (COPY, PRINT etc).

Syntax:

```
Help
```

Where:

No parameters required.

3.90.10. Job — Submit Job

This command submits the input specification (assumed to be OS/3 JCL) to the OS/3 run processor via the local reader queue. This command should only be issued if the OS/3 supervisor has been generated with input spooling (input spooling is automatically provided if remote spooling is specified).

Syntax:

```
Job  P1 [,P2] [,P3]
```

Specify the item to be submitted as a job stream from the following table:

Parameters P1, P2, P3	Description
File/elt [,type]	Input OS/3 Library element Default is source type "S".
group/name,E	TIP/30 edit buffer specification Pseudo type code "E" indicates TIP/30 Edit Buffer.

Example:

```
J  RUN/QWIKJOB,S
```

Submits an element named "QWIKJOB" from library "RUN" to the OS/3 reader and invokes the run processor symbiont to process it.

Additional Considerations:

The input is submitted on the assumption that it contains valid job control statements. In any case, once submitted, the job appears to the OS/3 system as if it was run by the system console — there is no provision for returning any completion status or to communicate with the job in any way.

The following options affect the JOB command:

Q Do not display any messages (Quiet)

S Scratch input edit buffer after processing.

Error Conditions:

The named element or edit buffer may not exist or the file cannot be accessed or the type may be invalid.

3.90.11. LIST — List Input at Terminal

The LIST command displays data on the terminal.

Syntax:

```
List  P1 [,P2] [,P3] [,printer]
```

Where:

Specify the item to be listed from the following table:

Parameters P1, P2, P3	Description
omitted	If parameters 1 through 3 are omitted, the terminal is used as an input device. Input is solicited line by line until MSG WAIT is pressed to signal end of input.
File/elt [,type]	Input OS/3 Library element Default is source type "S".
group/name,E	TIP/30 edit buffer specification Pseudo type code "E" indicates TIP/30 Edit Buffer.
d:ffffffff.eee	MS-DOS file specification: **d:** MS-DOS drive identifier **ffffffff** MS-DOS file name **.eee** MS-DOS file extension
AUXn,,,	Auxiliary device specification "n" is device number (1 through F) Some AUX devices are capable of write operations only and cannot be specified as an input device.

printer The destination printer (default is AUX0 — full screen display). Other possibilities are, for example, PRNTR, AUX1 or AUX1*BYP etc.

Example:

```
LIST  JCS,,D
```

Lists the directory of the file catalogued with logical file name "JCS".

Additional Considerations:

The following options affect the LIST command:

A	ASSEMBLER mode — use columns 1-72
C	COBOL mode — use columns 1-72 columns 1-6 set to spaces
Q	Do not display any messages (Quiet)
R	RPG mode — use columns 1-74 columns 1-5 set to spaces
S	Scratch input edit buffer

Error Conditions:

The named element may not exist or the file cannot be accessed or the type may be incorrect.

3.90.12. PRINT — Print Input

This command creates a printed display of the input specification at the site printer, an auxiliary print device or any print destination recognized by the TIPPRINT interface (see description of TIPPRINT in the documentation of the TIP/30 File Control System — FCS).

Unless inhibited by the appropriate option, output sent to the site printer is preceded by a separator (header) page to facilitate identification of the printout.

Each TLIB print request to the site printer is breakpointed by TIP/30 and may be printed by starting a burst mode output writer (ie: OS/3 operator command "PR BX,JOB=TIP30").

Syntax:

```
Print  P1 [,P2] [,P3] [,printer] [,hdr] [,case] [,plen] [,copies]
```

Where:

P1 [,P2] [,P3]

Specify the item to be printed from the following table:

Parameters P1, P2, P3	Description
omitted	If parameters 1 through 3 are omitted, the terminal is used as an input device. Input is solicited line by line until MSG WAIT is pressed to signal end of input.
File/elt [,type]	Input OS/3 Library element Default is source type "S".
group/name,E	TIP/30 edit buffer specification Pseudo type code "E" indicates TIP/30 Edit Buffer.
d:ffffffff.eee	MS-DOS file specification: d: MS-DOS drive identifier ffffffff MS-DOS file name .eee MS-DOS file extension
AUXn,,,	Auxiliary device specification "n" is device number (1 through F) Some AUX devices are capable of write operations only and cannot be specified as an input device.

printer The destination printer (default is the site printer PRNTR). Other possibilities are, for example, AUX1 or AUX1*BYP etc.

hdr YES/NO choice of a header (separator) page. Default is "N" if the destination is an AUX printer, otherwise, default is "Y".

case Choice of upper case translation. Default is translate to upper case ("U") if printer is the site printer, otherwise, default is no translation ("L").

plen The logical length of the page to be printed.

This specification is ignored for batch printers (spooled output). In such cases, the VFB information for the printer from the TIP/30 job control is honoured.

copies The number of copies to be generated; default is 1 copy.

Example:

```
PR  jcs/tip30,,aux1,n,U
```

Prints source element named "TIP30" from the library with catalogued logical file name "JCS" on the terminal's auxiliary printer. No separator page is to be printed and all alphabetic characters are to be translated to upper case.

Additional Considerations:

The declared format of a library element, an edit buffer or an MS-DOS file (ie: COBOL or Assembler or RPG etc) will cause the PRINT command to produce a printout that is more than a simple list of the lines.

Compiler page skip directives will be recognized:

- "/" in column 7 of a COBOL program.
- "EJECT" or "TITLE" in columns 10 through 14 of an assembler program.
- "/EJECT" in columns 7 through 12 of an RPG program.
- ";EJECT" in columns 1 through 6 of a source module.

The following options affect the PRINT command:

A ASSEMBLER mode — use columns 1-72

C COBOL mode — use columns 1-72
columns 1-6 set to spaces

L Print WITH line numbers

M Do not print heading lines (Minus)

N Do not print title page

Q Do not display any messages (Quiet)

R RPG mode — use columns 1-74
columns 1-5 set to spaces

S Scratch input edit buffer

Error Conditions:

The specified element or edit buffer was not found or the file could not be accessed or the type is invalid.

3.90.13. PUNCH — Create Punch Output

The PUNCH command creates a PUNCH file from a library element or edit buffer or library directory at the specified punch file (for example, LFD PUNCH).

Syntax:

```
PUnch  P1 [,P2] [,P3] [,punchname]
```

Where:

P1 [,P2] [,P3]

Specify the item to be "punched" from the following table:

Parameters P1, P2, P3	Description
omitted	If parameters 1 through 3 are omitted, the terminal is used as an input device. Input is solicited line by line until (MSG WAIT) is pressed to signal end of input.
File/elt [,type]	Input OS/3 Library element Default is source type "S".
group/name,E	TIP/30 edit buffer specification Pseudo type code "E" indicates TIP/30 Edit Buffer.
d:ffffffff.eee	MS-DOS file specification: **d:** MS-DOS drive identifier (A: C: etc) **ffffffff** MS-DOS file name **.eee** MS-DOS file extension
AUXn,,,	Auxiliary device specification "n" is device number (1 through F) Some AUX devices are capable of write operations only and cannot be specified as an input device.

punchname

The destination punch. Default is the site punch (PUNCH).

Example:

```
PUN  jcs/tip30
```

Punches source element named "TIP30" from the library with catalogued logical file name "JCS" to the site punch.

Additional Considerations:

The following options affect the PUNCH command:

Q	Do not display any messages (Quiet)
S	Scratch input edit buffer

Error Conditions:

The specified element or edit buffer was not found or the file could not be accessed or the type is invalid.

3.90.14. QUIT — End TLIB and LOGOFF

The QUIT command causes the TLIB program to discontinue prompting the user for more commands and terminates the TLIB program normally.

If the TLIB program was executing at stack level one (TLIB was NOT called by another program) the user will be logged off TIP/30.

Syntax:

```
Quit
```

Where:

No parameters are required.

3.90.15. RECOVER — Activate Previous Version

When an element of an OS/3 library is deleted, the module is not physically removed — the library index entry for it is marked as logically deleted.

The RECOVER or BACK command marks the currently active element as deleted, finds the previous version and reactivates its directory entry.

Elements that are marked as logically deleted are physically removed during a library compress operation that must be performed by the OS/3 batch utility program LIBS (the "pac" command).

The RECOVER or BACK command may be issued several times in succession to go back a number of versions (assuming they still exist).

If there is not a current active version of an element (for example, if the user accidentally deleted an element) then the user must first create a dummy current version before using the RECOVER or BACK command.

Syntax:

① `Back     file,elt [,type]`

② `REcover  file,elt [,type]`

Where:

file Logical file name (LFN) of the OS/3 library to process.

elt Name of the element to recover.

type Element type. Valid choices are S(ource), M(acro) or P(roc). Default "S".

The BACK command cannot process Object or Load type elements.

Example ①

```
TIP1?▶TLIB
TLIB(1)?▶BACK JCS/MYJOB
'TIP/30 Librarian' - Version = 4.0 (89/09/01)
JCS/MYJOB,S back to 89/01/12 14:33   Element comment
TLIB(1)?▶
```

Example ②

```
TIP1?▶RECOVER JCS MYJOB
'TIP/30 Librarian' - Version = 4.0 (89/09/01)
JCS/MYJOB,S back to 89/01/12 14:33   Element comment
TIP1?▶
```

Additional Considerations:

A successful RECOVER or BACK command reports the date and time stamp of the element that is reactivated. RECOVER may be used as an alternative spelling of BACK. RECOVER is also a clone transaction of TLIB (see "3.70. RECOVER — RECOVER Element" on page 3-227).

The following option affects the BACK command:

Q	Do not display any messages (Quiet)

Error Conditions:

The specified element may not currently exist, the file name may be invalid or it may not be possible to locate a "previous" version of the element.

3.90.16. SETON — Set TLIB Option On

The TLIB command SETON is used when TLIB is being executed interactively. In the interactive mode, TLIB prompts the terminal user for successive commands. If the desired command can benefit from a particular option, the only way the option can be turned ON is by first using the SETON command.

Of course, when TLIB is first invoked, one or more options may be set on via the command line. Once TLIB begins prompting for commands, the command line option field is no longer accessible — hence the need for an explicit command to manipulate options.

Syntax:

```
SETON opt, opt, opt ...
```

Where:

opt Each parameter represents up to 8 option characters that are to be set in the ON state. See "3.90.2. TLIB Options" on page 3-327 for a table of valid option characters.

There are seven parameters available after the command (SETON). TLIB allows a parameter to SETON to consist of one or more option characters.

The following two commands are identical:

SETON L,N,Q

SETON NLQ

Example:

```
TIP?▶TLIB
'TIP/30 Librarian' - Version = 4.0 (89/09/18)
TLIB(1)?▶seton l,n
ON=( L N )  OFF=( A C H I K M O Q R S T U X Y Z ? )   DISPLAY=(?)
TLIB(1)?▶print src/pay001,s
TLIB(1)?▶print src/pay002,s
TLIB(1)?▶print src/pay003,s
TLIB(1)?▶e
```

In this example, several Print commands are to be issued. Before the print commands, options L (print line numbers) and N (no header separator pages) are set on.

Note: The SETON command is often used in the construction of indirect input (a .IN file) for TLIB. See the description of redirected input in the documentation for the TIP/30 Program Control System — PCS.

3.90.17. SETOF — Set TLIB Option Off

The TLIB command SETOF is used when TLIB is being executed interactively. In the interactive mode, TLIB prompts the terminal user for successive commands. If the desired command requires a particular option to be OFF, the only way the option can be turned OFF is by first using the SETOF command.

Of course, when TLIB is first invoked, all options are initially OFF. Once TLIB begins prompting for commands, an option that was ON may be set OFF using this command.

Syntax:

```
SETOF opt, opt, opt ...
```

Where:

opt Each parameter represents up to 8 option characters that are to be set in the OFF state. See "3.90.2. TLIB Options" on page 3-327 for a table of valid option characters.

There are seven parameters available after the command (SETOF). TLIB allows a parameter to SETOF to consist of one or more option characters.

The following two commands are identical:

SETOF L,N,Q

SETOF NLQ

Example:

```
TIP?▶TLIB
'TIP/30 Librarian' - Version = 4.0 (89/09/18)
TLIB(1)?▶seton L,N
ON=( L N )  OFF=( A C H I K M O Q R S T U X Y Z ? )   DISPLAY=(?)
TLIB(1)?▶print src/pay001,s
TLIB(1)?▶print src/pay002,s
TLIB(1)?▶setof N
ON=( L )  OFF=( A C H I K M N O Q R S T U X Y Z ? )   DISPLAY=(?)
TLIB(1)?▶print src/pay010,s
```

In this example, several Print commands are to be issued. Before the print commands, options L (print line numbers) and N (no header separator pages) are set on. Before printing module PAY010, however, a header page is desired, so option N is set off.

Note: *The SETOF command is often used in the construction of indirect input (a .IN file) for TLIB. See the description of redirected input in the documentation for the TIP/30 Program Control System — PCS.*

3.91. TSTCOM — Communication Test Program

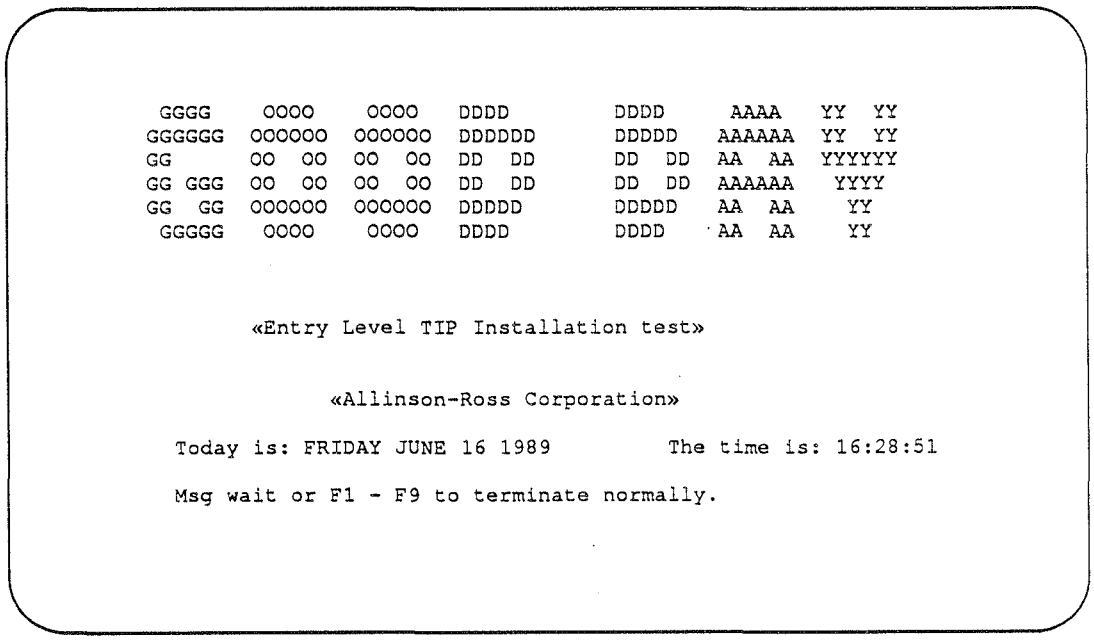
The TSTCOM program performs a test of the communication system by outputing a screen full of data and placing an "auto transmit" command sequence at the end of the output message. This results in a continuous output and input loop that can be used to simulate reasonably heavy communication traffic for the terminal (the output message is much larger than the input message which follows — this approximates a "typical" environment).

The program can be terminated by pressing the MSG WAIT key or F1 through F9 .

Pressing F10 causes the program to deliberately cause a program check by calling the TIPDUMP subroutine.

The TSTCOM program has no parameters. Enter only the transaction code.

Example of TSTCOM Output:



3.92. UNS — Unsolicited Console Keyin

The UNS transaction allows the user to submit an unsolicited "console" key in to an executing OS/3 job or symbiont. This is the same capability provided by the system operator console UNS command.

The UNS transaction is a clone of the transaction SYM (see "3.86. SYM — Schedule OS/3 Symbiont" on page 3-276). The UNS transaction is a separate transaction name to permit the system administrator to assign specific security constraints to this functionality.

The OS/3 UNS command syntax is documented in the

operation guide for your system.

Note: There is no provision for returning any completion status.

Syntax:

```
UNS  parameters
```

Where:

parameters

Whatever parameters that are required by the UNS command that is to be submitted.

Example of unsolicited key ins:

```
UNS TIPX STOP

UNS M2,S DO L,LIN1,NET1
```

3.93. UP — Set Line Up

This program enables the user to request that a communication line be set "up" by ICAM. TIP/30 will request (to ICAM) that the corresponding line be marked "up".

Syntax:

① `UP line-name`

② `UP term-name`

Where:

line-name The name of the line that is to be set up.

term-name A terminal on the line that is to be set up.

Example:

```
UP LIN3
```

Additional Considerations:

This program has no effect in a GLOBAL ICAM network (since GUST actually owns the lines and will not honour such requests).

3.94. USERS — Display User Directory

The USERS utility displays a list of valid TIP/30 user-ids on the terminal. The "comment" field from the user's TIP/30 catalogue record is also displayed (this field is usually used for descriptive information concerning the user — name, telephone extension etc.).

This information may be useful to users who wish to use the MAIL utility to send a message to an individual whose user-id is not known or immediately obvious.

To avoid reading all of the TIP/30 Catalogue every time this program is invoked, the USERS program maintains a list of user information in an edit buffer named TIPY/U$ERLI$T.

Syntax:

```
USERS[,C]    [username]
```

Where:

C	Option to indicate that the edit buffer containing the list of user information is to be re-created.
username	Optional parameter which specifies the users to list. Prefix notation may be used (ie: USERS *P). Default: all users will be listed.

Example of USERS Display

```
TF$USR01              TIP/30 User List Effective 89/01/10  $USERID$  $TID  14:49
                      ==================================
==User==  =======Identification=======   ==User==  =======Identification=======
A-R-C     ALLINSON-ROSS CORPORATION
ALLEN     DAVE ALLEN
BARD      WILLIAM SHAKESPEARE
SMITHY    JOHN J. SMITH

      F1:Refresh   F2:Forward   F3-Backward   F9:Print   Msg-Wait:Exit <_>
```

Pressing F9 allows you to print the complete user listing on a printer destination of your choice (you will be prompted for the desired printer destination).

Additional Considerations:

The listing is sorted into ascending order by user-id if the user has access to the transaction "SORT" (see "3.81. SORT — Sort Edit Buffer" on page 3-249). If the user does not have access to the SORT transaction, the list appears in random order (the physical order the user information appears in the TIP/30 catalogue).

If the TIPY/U$ERLI$T edit buffer does not exist or is out of date, USERS displays the message "Working - Please wait" and re-creates the edit buffer.

If this transaction is run in background (either as part of the system startup processing or manually), the program does not interact with a terminal, but simply constructs a fresh version of the edit buffer.

If the USERS program is run with a specific username as command line parameter one, the program reads the TIP/30 Catalogue directly for the single user's information.

3.95. WMI — Display User Information

The WMI (who am I?) program displays information on the terminal showing the user-id of the user logged on the terminal, the terminal name (as defined to the system), the current date and time, the version of both TIP/30 and OS/3 that is in use, and the features of TIP/30 that are configured.

Syntax:

```
WMI[/SLEV]
```

Where:

SLEV — Command line option to cause the WMI program to include security level information with the elective groups.

Example of WMI Output:

```
                                        17:10 THURSDAY DECEMBER 21 1989

            User-id: ALLINSON
             Groups:   1,ARC        1,ADMIN



           Security:    1
     Account number:
           Terminal: T001 - SPC (24,80)
   Tip Control Area: ARCTCA
          Site name: ARC TORONTO
     TIP/30 Version: 4.0 C40R0-000
       ICAM Network: NET1  LOCAP: TIP1
       OS/3 version: 13 00 S4

System attributes:
GLOBAL ICAM, DMS, DDP, SYSTEM DEBUG.




TIP?▶
```

3.96. ZZCLS — Close Online File(s)

The ZZCLS transaction is used to close online files. This transaction provides compatibility with the master terminal command of the same name that is available with IMS transaction processing systems.

The ZZCLS transaction permits up to 8 file names (or file name prefixes) to be specified after the transaction name. A comma or one or more spaces may be used as a delimiter between filenames.

Syntax:

```
ZZCLS file [,file] ...
```

Where:

file The LFD name of an online file that is to be closed (and made unavailable for online use). At least one file name or prefix must be specified.

Standard TIP/30 prefix notation may be used. For example, "*PAY" means all file names with an LFD name beginning with the three characters "PAY".

Additional Considerations:

The ZZCLS transaction program places the file name information into the Continuity Data Area (CDA) and calls the FCLOSE transaction. All of the facilities of the FCLOSE transaction apply except those facilities implemented via command line options. See "3.33. FCLOSE — Close File(s)" on page 3-93 for additional information.

Example:

```
ZZCLS PAYMAST,APMAST,*INV
```

3.97. ZZDWN — Disable Terminals

The ZZDWN transaction is an alternative spelling of the TIP/30 transaction program DISABLE (see "3.24. DISABLE — Disable Terminals" on page 3-80).

Syntax:

```
ZZDWN terms
```

Where:

terms Up to 8 positional parameters. Each parameter represents a terminal name to be disabled.

A terminal name may be specified by using standard prefix notation (ie: *T3 means all terminals that have a name beginning with the character string "T3").

The use of prefix notation might result in the specification of the terminal that is running the ZZDWN program. If this occurs, the ZZDWN program will not consider the running terminal as matching the prefix specification.

Example:

```
ZZDWN  T103
```

Error Conditions:

The ZZDWN program may report that a terminal name is not valid. This may occur because a terminal name was misspelled or the terminal is the one being used to run the ZZDWN program.

3.98. ZZOPN — Open Online File(s)

The ZZOPN transaction is used to open online files. This transaction provides compatibility with the master terminal command of the same name that is available with IMS transaction processing systems.

The ZZOPN transaction permits up to 8 file names (or file name prefixes) to be specified after the transaction name. A comma or one or more spaces may be used as a delimiter between filenames.

Syntax:

```
ZZOPN file [,file] ...
```

Where:

file The LFD name of an online file that is to be opened (and made available for online use). At least one file name or prefix must be specified.

Standard TIP/30 prefix notation may be used. For example, "*PAY" means all file names with an LFD name beginning with the three characters "PAY".

Additional Considerations:

The ZZOPN transaction program places the file name information into the Continuity Data Area (CDA) and calls the FOPEN transaction. All of the facilities of the FOPEN transaction apply except those facilities implemented via command line options. See "3.36. FOPEN — Open Online File(s)" on page 3-98 for additional information.

Example:

```
ZZOPN PAYMAST,APMAST,*INV
```

3.99. ZZPCH — Reload Program

The ZZPCH transaction program is an alternative name for the RELOAD transaction program (see "3.71. RELOAD — Reload Program" on page 3-228).

When an on-line program is recompiled and relinked, the programmer may use the ZZPCH transaction to force TIP/30 to "refresh" the load module before using it again.

This transaction is provided for compatibility with the terminal command of the same name that is available for users of Information Management System (IMS).

Syntax:

```
ZZPCH loadm
```

Where:

loadm The load module name (trailing zeroes need not be entered).

Example:

```
TIP?▶zzpch tt$mod
TT$MOD00 cleared from loadr table.
TT$MOD00 cleared from reentrant control table.
TT$MOD00 ver: 89/09/15 @ 10:32 (C) A.R.C  TIP/30 4.0          4051 bytes
TIP?▶
```

3.100. ZZUP — Enable Terminals

The ZZUP transaction is an alternative spelling of the TIP/30 transaction program ENABLE (see "3.30. ENABLE — Enable Terminal Input" on page 3-89).

Syntax:

```
ZZUP terms
```

Where:

terms Up to 8 positional parameters. Each parameter represents a terminal name to be enabled.

A terminal name may be specified by using standard prefix notation (ie: *T3 means all terminals that have a name beginning with the character string "T3").

The use of prefix notation might result in the specification of the terminal that is running the ZZUP program. If this occurs, the ZZUP program will not consider the running terminal as matching the prefix specification.

Example:

```
ZZUP  T103
```

Error Conditions:

The ZZUP program may report that a terminal name is not valid. This may occur because a terminal name was misspelled or the terminal is the one being used to run the ZZUP program.

Glossary

This section supplies working definitions of some of the common terms used in the TIP/30 documentation. The definitions are not intended to be rigorous; they are explanations within the context of the TIP/30 system.

A

ACK Acknowledge(ment). A signal indicating that error detection logic has failed.

ASCII American Standard Code for Information Interchange. A set of character representations that associates single byte binary values with external graphic characters. See also "byte" and "EBCDIC". The ASCII character set is typically used by communications hardware for data transmission.

asynchronous

Happening simultaneously but independently.

auxiliary device

A peripheral unit (such as a printer, diskette, or cassette) attached to a terminal.

B

Background

As in ... process. A background process is a TIP/30 online transaction that is running but not associated with a physical terminal.

Background processes are non-interactive programs.

batch Not interactive.

bi-synch Bi-synchronous; a communications protocol which implies that traffic is synchronized in both directions by acknowledgement messages.

bit bucket A mythical and cavernous receptacle which is provided by hardware manufacturers to hold any data which is deliberately or accidentally mislaid during data manipulation.

For example, digits to the right of the decimal place that are truncated by a move operation fall into the bit bucket.

bypass A terminal that has an identifiable polling address but typically has no keyboard or display screen. Bypass terminals are often utilized to perform printing operations since they do have memory and auxiliary device capability.

byte The smallest addressable unit of storage in memory. A byte is composed of 8 bits (binary digits). The value of a byte ranges from zero to 255 (decimal) or 0 to FF (hexadecimal). Each of the characters in the computer's character set (either ASCII or EBCDIC) may be stored in a byte using a unique representation from the 256 possible binary values that may be stored in a byte.

C

catalogue (OS/3)

A directory of file names and corresponding volume label location information (stored in file YCAT).

catalogue (TIP/30)

A directory of information about users, transaction programs, and online files.

CRT Literally, Cathode Ray Tube. Often used to refer to the display screen of a computer terminal.

cursor A current position marker on a CRT. Usually a blinking rectangle or underline character that reminds the user where the next character will appear on the screen.

D

Direct Access

A file organization technique that numbers fixed size records using integers from 1 to the highest record number.

Doubleword

On OS/3 hardware a doubleword is 8 consecutive bytes beginning on an address that is evenly divisible by 8 (the right most 3 bits of the address are zero).

An area is said to be "doubleword aligned" if it begins at an address that is evenly divisible by 8.

COBOL aligns all WORKING-STORAGE level 01 items on a doubleword boundary.

TIP/30 aligns all of the external work areas for a transaction program (PIB, CDA, MCS, WORKAREA) on a doubleword boundary.

dynamic file

A TIP/30 pseudo-file that has the characteristics of direct access.

May be created, manipulated and erased (scratched) on demand by TIP/30 transaction programs.

E

EBCDIC Extended Binary-Coded Decimal Interchange Code. A set of character representations that associates single byte binary values with external graphic characters. See also "byte" and "ASCII". The EBCDIC character set is typically used by the CPU for internal data representation.

edit buffer A particular type of TIP/30 dynamic file that is used by the TIP/30 text editors as a work space for editing.

element The name of a library member or module.

F

FCS File Control System. TIP/30 interface between programs and on-line files.

Foreground

As in ... process. A foreground process is a TIP/30 online transaction that is running at a physical terminal.

Fullword

On OS/3 hardware a fullword is 4 consecutive bytes beginning on an address that is evenly divisible by 4 (the right most 2 bits of the address are zero).

An area is said to be "fullword aligned" if it begins at an address that is evenly divisible by 4.

A fullword can be defined in COBOL by specifying a PICTURE of 9(6) through 9(9) COMP SYNC.

Function Key

A key on a UNISCOPE terminal keyboard (numbered F1 through F22) which signals the host computer when pressed. NO data is sent from the terminal.

H

Halfword

On OS/3 hardware a halfword is 2 consecutive bytes beginning on an address that is evenly divisible by 2 (the right most bit of the address is zero).

An area is said to be "halfword aligned" if it begins at an address that is evenly divisible by 2.

A halfword can be defined in COBOL by specifying a PICTURE of 9(1) through 9(4) COMP SYNC.

hardware The physical computer equipment.

hashing A technique of computing a key from a value. Typically used to map a large number of values onto a smaller set of values.

Host computer

The main computer; the computer which is running TIP/30.

I

IMS A Unisys software product that provides an execution environment for transaction programs.

IMS emulation

A facility of TIP/30 which enables a transaction program written to use the facilities of IMS to run under control of TIP/30 without change or recompilation.

index A collection of keys and associated location information that can be searched to locate an item with a given key.

interactive

Operating in "question and answer" mode.

An interactive program presents decisions for a user to make and acts according to the response.

ISAM Indexed Sequential Access Method. A file organization method that allows access to records either randomly by a single key or sequentially by a single key.

Records may be fixed length or variable length (in the Unisys OS/3 implementation).

K

key A portion of the data in a record which is used to index the record.

L

LFD The name of a file as stated in the Job Control information for the job which refers to the file.

LFN Logical File Name. The name by which a TIP/30 program refers to a file. The logical file name is associated with the LFD name of the file by TIP/30 catalogue information.

M

MIRAM Multiple Indexed Random Access Method. File organization method that is similar to ISAM with the exception that there may be from one to five keys.

MSG-WAIT

Key on UNISCOPE terminals that signals the host computer when pressed (NO data is sent from the terminal).

multi thread

A number of transactions concurrently sharing resources.

N

native mode

A program that uses TIP/30 facilities that is NOT running under the control of the TIP/30 IMS/90 emulator is said to be running in this mode.

NAK Negative acknowledgement.

O

OS/3 Operating System 3. The control software supplied by Unisys for use on Series 90 and System 80 machines.

P

prefix notation

A notation convention adopted by most TIP/30 utilities to allow selection by prefix.

Eg: "*ABC" means all names with prefix "ABC"

Eg: "!XYZ" means all names NOT with prefix "XYZ"

An imbedded "?" or "." character implies that the corresponding position may be occupied by any character (for example: *A??B matches A12B or AXYB).

S

single thread

A method of transaction processing which allows one transaction to monopolize resources until completion of the transaction.

SOE (character). Start Of Entry character. On UNISCOPE terminals a character (shaped like a pennant blowing from left to right) which marks the left most boundary of data to be transmitted to the host computer.

Example: ▸

software The programs which control the operation of the hardware or other (application) programs.

T

TPS Transaction Platform System. A significant subset of the TIP/30 system that is included with System 80 Model 7E processors and available as a priced item for other OS/3 hardware platforms.

transaction

A program that executes under the control of TIP/30.

TIP See TIP/30.

TIP/30 Transaction Interface Processor — a system software product of Allinson-Ross Corporation.

U

unsolicited

As in ... message. A message sent to a terminal that is not necessarily a response to a previous input message.

In effect, a message sent gratuitously by another process in the system which arrives unexpectedly.

An unsolicited message is queued by ICAM until such time as the terminal operator presses the MSG-WAIT key (at which time ICAM will display the message on the terminal).

X

XMIT Transmit. A key on UNISCOPE terminals that sends data from the CRT to the host computer.

Index

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

W

X

Z

Help Us To Help You

Publication Title

Form Number Date

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments

Name

Title Company

Address (Street, City, State, Zip)

Telephone Number

Help Us To Help You

Publication Title

Form Number Date

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments

Name

Title Company

Address (Street, City, State, Zip)

Telephone Number

Help Us To Help You

Publication Title

Form Number Date

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments

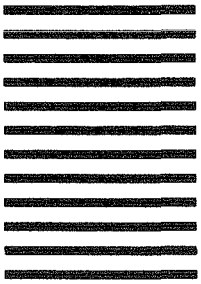
Name

Title Company

Address (Street, City, State, Zip)

Telephone Number

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



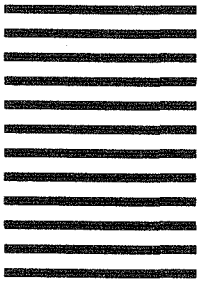
BUSINESS REPLY MAIL

First Class Permit No. 21 Blue Bell, PA

Postage Will Be Paid By Addressee

Unisys Corporation
OS/3 Systems Product Information Development
PO Box 500 - E5-114
Blue Bell, PA 19422-9990

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



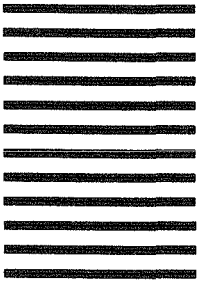
BUSINESS REPLY MAIL

First Class Permit No. 21 Blue Bell, PA

Postage Will Be Paid By Addressee

Unisys Corporation
OS/3 Systems Product Information Development
PO Box 500 - E5-114
Blue Bell, PA 19422-9990

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

First Class Permit No. 21 Blue Bell, PA

Postage Will Be Paid By Addressee

Unisys Corporation
OS/3 Systems Product Information Development
PO Box 500 - E5-114
Blue Bell, PA 19422-9990