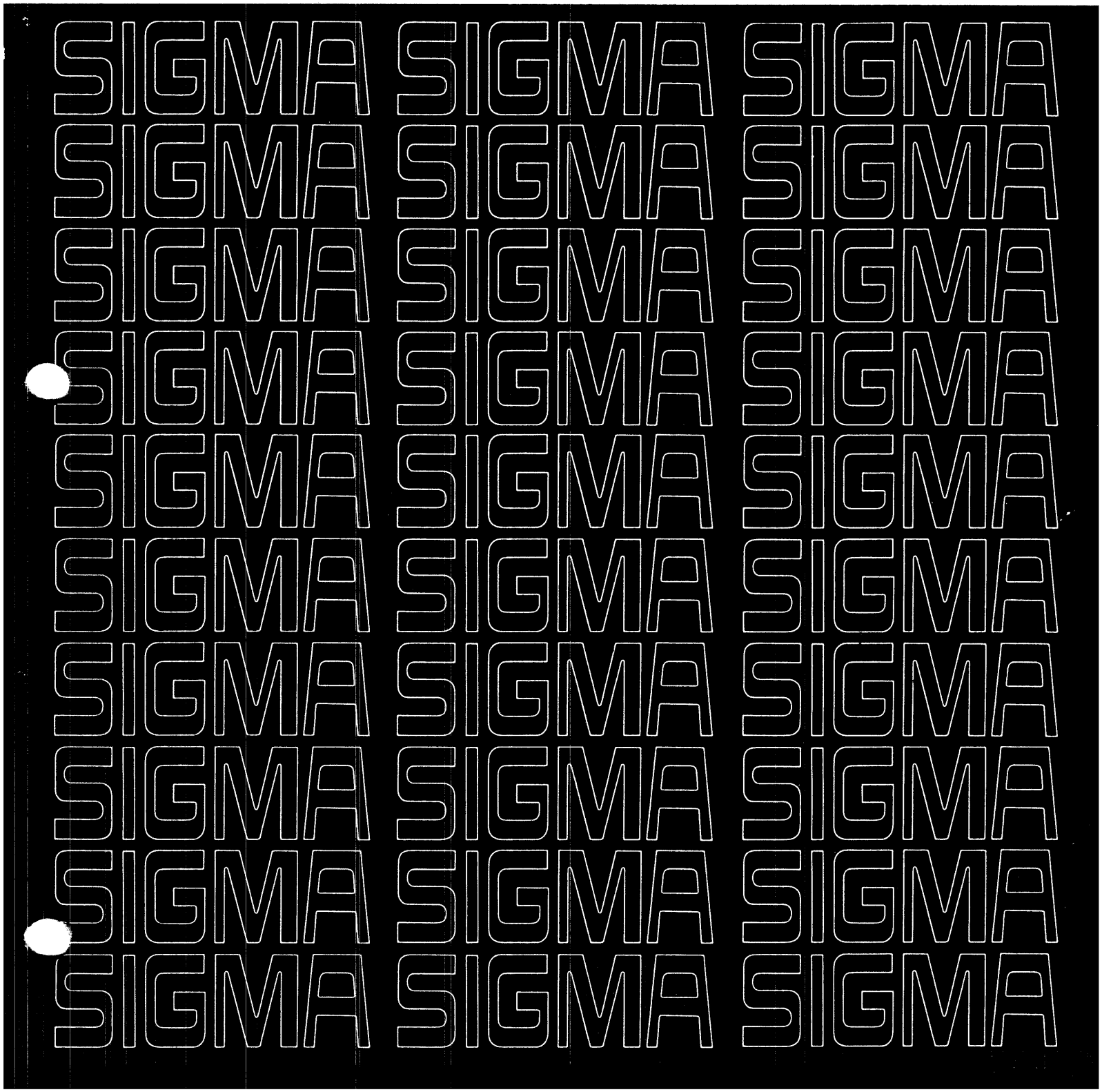
Xerox Data Systems

XEROX

Xerox Universal Time-Sharing System (UTS)

Sigma 6/7/9 Computers

Time-Sharing Reference Manual



Xerox Data Systems

701 South Aviation Boulevard
El Segundo, California 90245
213 679-4511

XEROX

Xerox Universal Time-Sharing System (UTS)

Sigma 6/7/9 Computers

Time-Sharing

Reference Manual

90 09 07C

November 1971

Price: $5.00

Printed in U.S.A.

REVISION

This publication is a revision of the Xerox Universal Time-Sharing System (UTS)/TS Reference Manual for Sigma 7 Computers, Publication Number 90 09 07B (dated February 1971). This revision documents the B00 release of the system. A change in text from that of the previous manual is indicated by a vertical line in the margin of the page. Chapter 6 (Edit) and Chapter 7 (Delta) are new chapters in this manual and are rewritten versions of the Edit Reference Manual, 90 16 33 and the Delta Reference Manual, 90 16 34.

RELATED PUBLICATIONS

Title	Publication No.
Xerox Sigma 6 Computer/Reference Manual	90 17 13
Xerox Sigma 7 Computer/Reference Manual	90 09 50
Xerox Sigma 9 Computer/Reference Manual	90 17 33
Xerox Universal Time-Sharing System (UTS)/OPS Reference Manual	90 16 75
Xerox Universal Time-Sharing System (UTS)/SM Reference Manual	90 16 74
Xerox Universal Time-Sharing System (UTS)/BP Reference Manual	90 17 64
Xerox Universal Time-Sharing System (UTS)/TS User's Guide	90 16 92
Xerox BASIC/LN,OPS Reference Manual	90 15 46
Xerox Meta-Symbol/LN,OPS Reference Manual	90 09 52
Xerox Extended FORTRAN IV/LN Reference Manual	90 09 56
Xerox Extended FORTRAN IV/OPS Reference Manual	90 11 43
Xerox FORTRAN Debug Package (FDP)/Reference Manual	90 16 77
Xerox ANS COBOL/LN Reference Manual	90 15 00
Xerox ANS COBOL/OPS Reference Manual	90 15 01
Xerox Manage/Reference Manual	90 16 10
Xerox Sort-Merge/Reference Manual	90 11 99
Xerox Functional Mathematical Programming System (FMPS)/Reference Manual	90 16 09
Xerox SL-1/Reference Manual	90 16 76
Xerox 1400 Series Simulator/Reference Manual	90 15 02

Manual Content Codes: BP – batch processing, LN – language, OPS – operations, RBP – remote batch processing, RT – real time, SM – system management, TS – time-sharing, UT – utilities.

CONTENTS

APPENDIXES

FIGURES

TABLES

GLOSSARY

address resolution code: a 2-bit code that specifies whether an associated address is to be used as a byte address or is to be converted (by truncating low order bits) to a halfword, word, or doubleword address.

batch job: a job that is submitted to the batch job stream through the card reader, or through an on-line terminal (using the BATCH command).

binary input: input from the device to which the BI (binary input) operational label is assigned.

conflicting reference: a reference to a symbolic name that has more than one definition.

control command: any control message other than a key-in. A control command may be input via any device to which the system command input function has been assigned (normally a card reader).

control key-in: a control message of the type that must be input from the operator's console.

control message: any message received by the Monitor that is either a control command or a control key-in.

data control block (DCB): a table in the user program that contains the information used by the Monitor in the performance of an I/O operation.

external reference: a reference to a declared symbolic name that is not defined within the object module in which the reference occurs. An external reference can be satisfied only if the referenced name is defined by an external load item in another object module.

file extension: a convention that is used when certain system output DCBs are opened. Use of this convention causes the file (RAD, tape, disk pack, etc.) connected to the DCB to be positioned to a point just following the last record in the file. Thus, when additional output is produced through the DCB, it is added to the previous contents of the file, thereby extending the file.

function parameter table (FPT): a table through which a user's program communicates with a Monitor function (such as an I/O function).

ghost job: a job that is initiated by the Monitor, the operator, or a program that is neither a batch nor an on-line program.

global symbol: a symbolic name that is defined in one program module and referenced in another.

GO file: a temporary RAD storage file created, for example, from relocatable object modules formed by a processor. Such modules may be retrieved by use of a LOAD or RUN control command in batch mode or a dollar sign in on-line mode.

internal symbol: a symbolic name that is defined and referenced in the same program module.

job information table (JIT): a table associated with each active job. The table contains accounting, memory mapping, swapping, terminal DCB (M:UC), and temporary Monitor information.

job step: a subunit of job processing such as compilation, assembly, loading, or execution. Information from certain commands (JOB, LIMIT, and ASSIGN) and all temporary files created during a job step are carried from one job step to the next but the steps are otherwise independent.

key: a data item consisting of 1-31 alphanumeric characters that uniquely identifies a record.

key-in: information entered by the operator via a keyboard.

linking loader: a program that is capable of linking and loading one or more relocatable object modules and load modules.

load map: a listing of loader output showing the location or value of all global symbols entering into the load. Also shown are symbols that are not defined or have multiple definitions.

load module (LM): an executable program formed by the linking loader, using relocatable object modules (ROMs) and/or load modules (LMs) as source information.

logical device: a peripheral device that is represented in a program by an operational label (e.g., BI or PO) rather than by specific physical device name.

monitor: a program that supervises the processing, loading, and execution of other programs.

object language: the standard binary language in which the output of a processor is expressed.

object module: the series of records containing the load information pertaining to a single program or subprogram (i.e., from the beginning to the end). Object modules serve as input to the loader.

on-line job: a job that is submitted through an on-line terminal by a command other than the BATCH command.

operational label: a symbolic name used to identify a logical system device.

option: an elective operand in a control command, procedure call, or on-line command, or an elective parameter in a Function Parameter Table.

parameter presence indicator: a bit in word 1 of a Function Parameter Table that indicates whether a particular parameter word is present in the remainder of the table.

physical device: a peripheral device that is referred to by a name specifying the device type, I/O channel, and device number (also see "logical device").

prompt character: a character that is sent to the terminal by an on-line language processor to indicate that the next line of input may be entered.

public library: a set of library routines declared, at System Make time, to be public (i.e., to be used in common by all concurrent users).

reentrant: an attribute of a program that allows the program to be shared by several users concurrently. Shared processors in UTS are map reentrant. That is, each instance of execution of a single copy of the program's instructions has a separate copy of the execution data.

relative allocation: allocation of virtual memory to a user program starting with the first unallocated page available.

relocatable object module (ROM): a program, or subprogram, generated by a processor such as Meta-Symbol or FORTRAN (in XDS Sigma 7 object language).

resident program: a program that has been loaded into a specific area of core memory.

secondary storage: any rapid-access storage medium other than core memory (e.g., RAD storage).

shared processor: a program (e.g., FORTRAN) that is shared by all concurrent users. Shared processors must be established in UTS by SYSGEN.

source language: a language used to prepare a source program suitable for processing by an assembler or compiler.

special shared processor: a shared processor that may be in core memory concurrently with the user's program (e.g., Delta or TEL).

specific allocation: allocation of a specific page of unallocated virtual memory to a user program.

static core module: a program module that is in core memory but is not being executed.

symbiont: a Monitor routine that transfers information between RAD storage and a peripheral device independent of and concurrent with job processing.

symbolic input: input from the device to which the SI (symbolic input) operational label is assigned.

symbolic name: an identifier that is associated with some particular source program statement or item so that symbolic references may be made to it even though its value may be subject to redefinition.

system library: a group of standard routines in object-language format, any of which may be incorporated in a program being formed.

system register: a register used by the Monitor to communicate information that may be of use to the user program (e.g., error codes). System registers SR1, SR2, SR3, and SR4 are current general registers 8, 9, 10, and 11, respectively.

task control block (TCB): a table of program control information built by the loader when a load module is formed. The TCB is part of the load module and contains the data required to allow reentry of library routines during program execution or to allow a synchronous entry to the program in cases of traps, breaks, etc. The TCB is program associated and not task associated.

unsatisfied reference: a symbolic name that has been referenced but not defined.

1. INTRODUCTION

DEFINITION OF UTS

The Universal Time-Sharing System (UTS) is a comprehensive operating system designed for use with Sigma 6/7/9 computers. It provides a time-shared computing service with a Sigma 6, 7, or 9 computer as the central computer and up to 128 on-line terminals. In addition to time-sharing services, UTS provides local batch service.

TIME-SHARING SERVICE

There are three general categories of time-sharing service provided to on-line users. They are on-line file management, on-line program execution and debugging, and on-line entry of jobs into the batch job stream. The processors that provide these services are listed in Table 1 and are discussed in more detail in the following paragraphs.

TERMINAL EXECUTIVE LANGUAGE

The terminal Executive Language (TEL) is the principal terminal language for UTS. Most activities associated with FORTRAN and assembly language programming can be carried out directly in TEL through requests that take the form of single-line commands and declarations. These activities include such major operations as composing programs and other bodies of text, compiling and assembling programs, linking object programs, initiating execution, and debugging programs. They also include such minor operations as checkpointing on-line sessions, determining program status, and setting simulated tab stops. (Reference: Chapter 3.)

Table 1. On-Line User Processors

Processor	Function
TEL	Executive language control of all terminal activities.
EDIT	Composition and modification of programs and other bodies of text.
FORT4	Compilation of Xerox Extended FORTRAN IV programs.
META	Assembly of Meta-Symbol programs.
BASIC	Compilation and execution of programs or direct statements written in an extended BASIC language.
FDP	Debugging of Xerox Extended FORTRAN IV programs.
DELTA	Debugging of programs at the machine language level.
PCL	Transfer (and conversion) of data between peripheral devices.
LINK	Linkage of programs for execution

EDIT

The Edit processor is a line-at-a-time context editor designed for on-line creation, modification, and handling of programs and other bodies of information. All Edit data is stored on RAD or disk pack storage in a keyed file structure of sequence-numbered variable length records. This structure permits Edit to directly access each line or record of data.

Edit functions are controlled through single-line commands supplied by the user. The command language provides for insertion, deletion, reordering, and replacement of lines or groups of lines of text. It also provides for selective printing, renumbering records, and context editing operations of matching, moving, and substituting line-by-line within a specified range of text lines. File maintenance commands are also provided to allow the user to build, copy, and delete whole files of text lines. (Reference: Chapter 6.)

XEROX EXTENDED FORTRAN IV

The Xerox Extended FORTRAN IV language processor (FORT4) consists of a comprehensive algebraic programming language, a compiler, and a large library of subroutines. The language is a superset of most available FORTRAN languages, containing many extended language features to facilitate program development and checkout. The compiler is designed to produce very efficient object code, thus reducing execution time and core requirements, and to generate extensive diagnostics to reduce debugging time. The library contains over 180 subprograms and is available in a reentrant version. Both the compiler and runtime library for object programs are reentrant programs that are shared among all concurrent users to improve the utilization of the critical core resources.

The principal features of Xerox Extended FORTRAN IV are as follows:

Extended language features to reduce programming effort and increase range of applications.

Extensive meaningful diagnostics to minimize debugging time.

In-line assembly language code to reduce execution time of critical parts of the program.

Overlay organization for minimal core memory utilization.

Compiler produced reentrant programs.

Full use of UTS features.

Availability of reentrant version of library.

(Reference: Extended FORTRAN IV/LN Reference Manual, 90 09 56 and Extended FORTRAN IV/OPS Reference Manual, 90 11 43.)

META-SYMBOL

Meta-Symbol is a procedure-oriented macro assembler that provides services available in sophisticated macro assemblers and has special features that permit the user to execute dynamic control over the parametric environment of assembly. Meta-Symbol's highly flexible assembly language gives users full use of the available Sigma hardware capabilities.

Under UTS, Meta-Symbol may be used in batch or on-line mode. In on-line mode, the assembler allows programs to be assembled and executed on-line but does not allow conversational interaction.

One of Meta-Symbol's features is a highly flexible list definition and manipulation capability. Lists and list elements may be conveniently redefined, thus changing the value of a given element.

Another Meta-Symbol feature is the macro capability. XDS uses the term "procedure" to emphasize the highly sophisticated and flexible nature of this macro capability. Procedures are assembly-time subroutines that provide the user with an extensive function capability. Procedure definitions, reference, and recursions may be nested up to 32 levels.

Meta-Symbol also has an extensive set of operators to facilitate the use of logical and arithmetic expressions. These operators facilitate the parametric coding capabilities available with Meta-Symbol (parametric programming allows for dynamic specification of both "if" and "how" a given statement or set of statements is to be assembled).

Users are also provided with an extensive set of directives. These directives, which are commands intrinsic to the assembly, fall into three classes:

1. Directives that involve manipulation of symbols and are unconditionally executed.

2. Directives that allow parametric programming.

3. Directives that do not allow parametric programming.

Intrinsic functions are also included in Meta-Symbol. These give the user the ability to obtain information on both the structure and content of assembly time constructs. For example, the user can acquire information on the length of a certain list. He can inquire about a specific symbol and whether it occurs in a procedure reference. (Reference: Meta-Symbol/LN, OPS Reference Manual, 90 09 54.)

BASIC

BASIC is a compiler and programming language similar to Dartmouth BASIC. It is, by design, easy to teach, learn, and use. It allows individuals with little or no programming experience to create, debug, and execute programs via an on-line terminal. Such programs are usually small to medium size, predominantly arithmetic applications.

BASIC is designed primarily for on-line program development and execution, or on-line development and batch execution. In addition, programs may be developed and executed in batch mode.

BASIC provides two user modes of operation. The editing mode is used for creating and modifying programs. The compilation/execution mode is used for running completed programs. This arrangement simplifies and speeds up the program development cycle.

BASIC statements may be entered via a terminal and immediately executed. During execution, programs may be investigated for loop detection, snapshots of variables may be obtained, values of variables may be changed, flow of execution may be rerouted, and so on. This unique capability also allows an on-line terminal to be used as a "super" desk calculator.

At compile and execute time, the user may specify an array dimension check. In the safe mode, statements are checked to verify that they do not reference an array beyond its dimensions. In the fast mode, this time consuming check is not made. The safe mode is used during checkout; the fast mode is used when the program reaches the production state, to speed up execution.

BASIC provides an image statement that uses a "picture" of the desired output format to perform editing. It also has TAB capability and a precision option to indicate the number of significant digits (6 to 16) to be printed.

BASIC also has an easy-to-use feature allowing the user to read, write, and compare variable alphanumeric data. This is particularly important for conversational input processing.

Chaining permits one BASIC program to call upon another for compilation and execution without user intervention. Thus, programs that would exceed user core space may be sequenced and overlay techniques may be employed via the chaining facility. (Reference: BASIC/Reference Manual, 90 15 46.)

FORTRAN DEBUG PACKAGE

The FORTRAN debug package (FDP) is made up of special library routines that are called by XDS Extended FORTRAN IV object programs compiled in the debug mode. These routines interact with the program to detect, diagnose, and in many cases, temporarily repair program errors.

The debugger can be used in batch and on-line mode. An extensive set of debugging commands are available in both cases. In batch operation, the debugging commands are included in the source input and are used by the debugger during execution of the program. In on-line operation, the debugging commands are entered through the terminal keyboard when requested by the debugger. Such requests are made when execution starts or restarts and for all execution stops in which the debugger has control. The debugger normally has control of such stops.

In addition to the debugging commands, the debugger has a few automatic debugging features. One of these features is the automatic comparison of standard calling and receiving sequence arguments for type compatibility. When applicable, the number of arguments in the standard calling sequence is checked for equality with the number of dummies in the receiving sequence. These calling and receiving arguments are also tested for protection conflicts. Another automatic feature is the testing of subprogram dummy storage attempts, to determine if they violate the protection of the calling argument. (Reference: FORTRAN Debugger/ Reference Manual, 90 16 77.)

DELTA

Although Delta is designed to aid in the debugging of programs at the assembly-language or machine-language levels, it may be used to debug FORTRAN, COBOL, or any other program. It is designed and interfaced with UTS in such a way that it may be called in to aid debugging at any time (even after a program has been loaded).

Delta operates on object programs and tables of internal and global symbols used by the programs but does not require the tables to be present. With or without the symbol tables, Delta recognizes computer instruction mnemonic codes and can assemble machine-language programs on an instruction-by-instruction basis. The main purpose of Delta, however, is to facilitate the activities of debugging by

1. Examining, inserting, and modifying program elements such as instructions, numeric values, coded information (i.e., data in all its representations and formats).

2. Controlling execution, including the insertion of break points into a program and requests for breaks on changes in elements of data.

3. Tracing execution by displaying information at designated points in a program.

4. Searching programs and data for simple elements or elements within a hierarchy.

To assist the first activity, UTS assemblers and compilers include information identifying the type of data each symbol in the symbol table represents. The type of data includes symbolic instructions, decimal integers, floating-point values, single and double precision values, EBCDIC encoded information, and other types. (Reference: Chapter 7.)

PERIPHERAL CONVERSION LANGUAGE

The Peripheral Conversion Language (PCL) is a utility subsystem designed for operation in a batch or on-line environment under UTS. It provides for information movement among card devices, line printers, on-line terminals, magnetic tape devices, disk packs, and RAD storage.

PCL is controlled by single-line commands supplied through on-line terminal input, through a file containing PCL commands, or through command card input in the job stream. The command language provides for single or multiple file transfers with options for selecting, sequencing, formatting, and converting data records. Additional file maintenance and utility commands are provided. (Reference: Chapter 5.)

LINK

Link is a linking loader that constructs a single entity called a load module, which is an executable program formed from relocatable object modules. Link is a one-pass linking loader that makes full use of mapping hardware. It is not an overlay loader. If the need for an overlay loader exists, the overlay loader (Load) must be called by entering the job in the batch stream. (Reference: Chapter 8.)

SUPER

Super[†] gives system management control over the entry of users and the privileges extended to users. Through the use of Super commands, a system manager may add and delete users, specify how much core and RAD storage space a user will have, and control the use of central site magnetic tape units, symbiont printers and punches. He may also grant certain users, say system programmers, special privileges such as the privilege of examining, accessing, and changing the Monitor. (Reference: UTS/SM Reference Manual, 90 16 74.)

CONTROL

The Control processor[†] provides on-line control over system performance. UTS has a number of performance measures built directly into the system. Commands of the Control processor enable system management to display these measurements and to "tune" the system as needed. (Reference: UTS/SM Reference Manual, 90 16 74.)

RATES

The Rates processor[†] allows the system manager to set relative charge weights on the utilization of system services. (Reference: UTS/SM Reference Manual, 90 16 74.)

[†]These processors are system management processors and are not available to other on-line terminal users.

FILL

The Fill processor[t] performs three basic file maintenance functions:

1. It copies files from disk to tape as a backup.
2. It restores files from tape to disk.
3. It deletes files from disk.

(Reference: UTS/SM Reference Manual, 90 16 74.)

UTSPM

The UTS Performance Monitor (UTSPM)[t] displays and collects performance data on a running system and produces snapshot files to be displayed by the report generator Summary. (Reference: UTS/SM Reference Manual, 90 16 74.)

SUMMARY

The Summary processor[t] provides a global view of UTS performance by formatting and displaying the statistical data collected by UTSPM. (Reference: UTS/SM Reference Manual, 90 16 74.)

DRSP

DRSP[t] (Dynamic Replacement of Shared Processors) enables the system manager to dynamically add, replace, or delete shared processors and Monitor overlays. He may do this during normal system operations with other users on the system. (Reference: UTS/SM Reference Manual, 90 16 74.)

SYSGEN

SYSGEN[t] is made up of several processors. These processors are used to generate a variety of UTS systems that are tailored to the specific requirements of an installation. (Reference: UTS/SM Reference Manual, 90 16 74.)

ANLZ

ANLZ[t] (Analyze) provides the system programmer with a means of examining and analyzing the contents of dumps taken during system recovery. (Reference: UTS/SM Reference Manual, 90 16 74.)

MANAGE

Manage[tt] is a generalized file management system. It is designed to allow decision makers to make use of the computer to generate and update files, retrieve useful data, and generate reports without having a knowledge of programming. (Reference: Manage/Reference Manual, 90 16 10.)

BATCH SERVICE

Batch processing facilities are described in UTS/BP Reference Manual, 90 17 64. Although some facilities and processors are reserved for on-line use and others for batch use, the two classes of service are complementary. Generally speaking, anything that can be done in batch mode can be done on-line, although sometimes in a curtailed manner. In particular, compilers and assemblers are compatible across the two classes of service at source and relocatable levels. For example,

1. Processors for Extended FORTRAN IV and Meta-Symbol are available both in batch and on-line mode.
2. Programs compiled or assembled in batch can be linked with those produced on-line and can be run and debugged on-line.
3. Programs compiled or assembled on-line can be linked and run in batch mode.

(Reference: UTS/BP Reference Manual, 90 17 64.)

---

[t] These processors are system management processors and are not available to other on-line terminal users.

[tt] This processor is made available by XDS on an optional basis. It will be provided only to those users who execute a License Agreement for each applicable Sigma installation.

2. TERMINAL OPERATIONS

INTRODUCTION

The following types of on-line terminals may be used with UTS:

XDS Model 7015 Keyboard/Printer

Teletype® Models 33, 35, and 37

IBM 2741 Terminals

The terminal operations described in this chapter apply primarily to Teletype and XDS Model 7015 terminals (see Figure 1) and to the Terminal Executive Language (TEL). Operations that are unique for 2741 terminals are delineated at the end of the chapter. Terminal communication services to user programs are discussed in Chapter 10.

Seven facets of terminal operations are described in this chapter. They are

1. Initiating and ending on-line sessions.
2. Typing lines.
3. Typing commands.
4. Detecting and reporting errors.
5. Interrupting UTS.
6. Paper tape input.
7. 2741 and Teletype differences.

CONVENTIONS

A number of conventions are used in command specifications and examples throughout the remainder of this manual. These conventions are listed below.

BRACES { }

Braces enclose required alternatives.

BRACKETS [ ]

Brackets enclose optional parameters.

COMMAND TERMINATOR (RET) (LF)

These symbols indicate that a carriage return (RET) or line feed (LF) character has been sent to signal the end of a command.

ELLIPSIS ... or ⋮

The ellipsis indicates that parameters (...) or commands (⋮) have been omitted.

ESCAPE KEY (ESC)

The symbol (ESC) indicates that an ESC character has been sent.

α^{C} α^{S} α^{CS}

The superscript is used with the letters of terminal keys to indicate a combination of keys. For example L^{C} indicates a control shift (CONTROL and L keys) and L^{CS} indicates a control and case shift (CONTROL, SHIFT, and L keys).

UNDERSCORE

All terminal output received from TEL or a subsystem is underscored. Terminal input is not.

INITIATING AND ENDING ON-LINE SESSIONS

An on-line user must establish a connection with UTS and identify himself properly before he can use TEL or any of its subsystems. When a connection with UTS is established, UTS responds by typing

UTS AT YOUR SERVICE

ON AT (time and date)

LOGON PLEASE:

UTS then waits for user account, name, and password (optional) to be entered. The name must be from one to twelve characters in length; account and password must be from one to eight characters. Any of the following characters may be used in user account, name, and password:

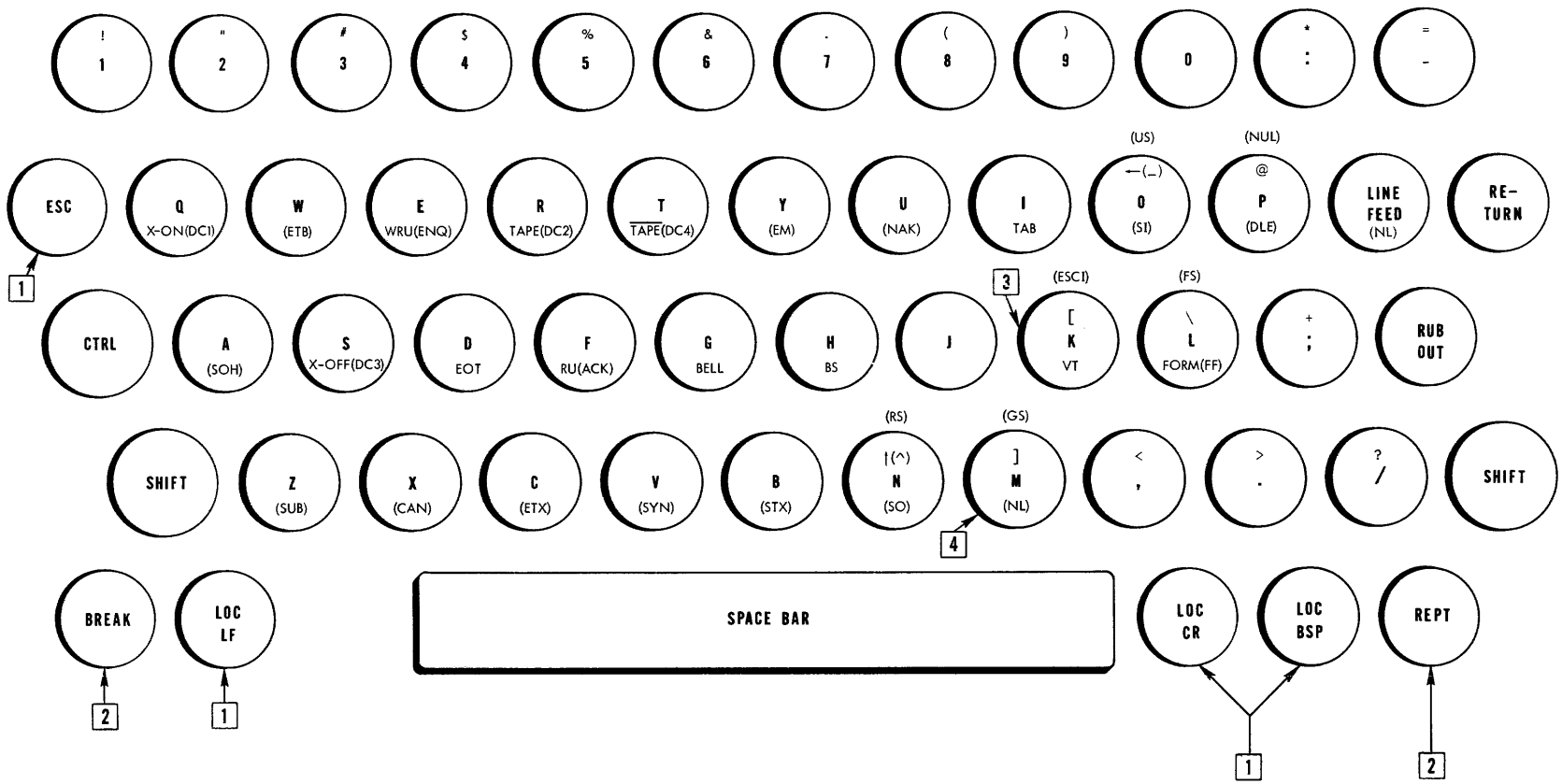
A-Z a-z 0-9 _ $ * % : # @ - backspace

Underscores count as characters and print as left-facing arrows (←). Commas are used as separators. After name (or password) is entered, the RETURN or LINE FEED key is depressed to return the carriage to the left margin of the next line and to deliver the line to UTS for examination.

For terminals operated in full-duplex mode, character echoing by the system is normally on but can be turned off (e.g., to suppress printing of passwords or other security-related information) by striking the (ESC) E keys. Striking the (ESC) E keys a second time turns echoing back on. For terminal units operated in half-duplex mode, character echoing by the system must be turned off, as above, to suppress duplicate printing of characters.

If the identification is valid and consistent with UTS records, TEL types its prompt character (!) at the left margin of the top line of the next page and then awaits the first command. If automatic association of a program or processor is

®Registered Trademark of the Teletype Corporation.



[1] These keys are missing on some models.

[2] This key is positioned elsewhere on some models.

[3] This is interpreted as | (OR) on Model 7015.

[4] This is interpreted as ¬ (NOT) on Model 7015.

Characters obtained by depressing the SHIFT key are shown at the top of the key and characters obtained by depressing the CTRL key are shown at the bottom of the key. Characters obtained by depressing both SHIFT and CTRL keys are shown above the key. On the actual keyboard, all unparenthesized forms appear at the top of the key.

Figure 1. Model 33 Teletype Terminal Keyboard

specified in the user's log-on record, control passes to that program instead of TEL for identification and command request. UTS sends an error message to the terminal and repeats the log-on request if the identification is garbled or otherwise invalid. The error messages are

EH? (Preceded by a repeat of the input for hardware debugging purposes.)

ACCOUNT/ID account/id?

PASSWORD?

It may not always be possible to log on. If an error prevents the reading of the log-on file, the message UNRECOVERABLE I/O ON RAD, or ABNORMAL ERROR ON LOGON FILE will be typed. Whenever the user is unable to log on, he may start over by striking the BREAK key and trying again. The system tries five times to log each user on before dismissing him.

Following a successful log-on, if the user has – in a previous session – exhausted his allocated permanent file storage space, he receives the following message:

FILE STORAGE LIMIT EXCEEDED

This means that no file storing operations can be performed until the user deletes one or more of his files.

If a MAILBOX file (a message file) exists at log-on time, the message CHECK DC/MAILBOX will appear. This MAILBOX file can be examined by copying it to the terminal as follows:

!COPY MAILBOX TO ME

(The underscored exclamation mark is the "prompt character" issued by TEL.)

The password in the log-on file can be set or changed at any time by typing PASSWORD xxxx, where xxxx is a character string from one to eight characters in length. (See Chapter 3.) Characters that may be used in a password command are as specified above.

If the PASSWORD command is used without specifying a password (xxxx), a password is no longer required for log-on.

An on-line session is ended by entering the OFF command. UTS sends the following use accounting information to the terminal when a user logs off:

CPU=m.mmmm CON=n:mm INT=nn CHG=xxxx

CPU time is expressed in minutes and ten-thousandths of a minute. Terminal time (CON) is expressed in hours and minutes. INT is the number of terminal interactions during the on-line session. CHG is the total number of charge units for the on-line session. (Reference: Chapter 4, UTS/SM Reference Manual, 90 16 74.)

TYPING LINES

The rules governing the typing of lines are concerned with operations such as erasing characters or lines, terminating lines, pagination, tabbing, and so on. These rules are common to TEL, all subsystems, and programs that carry on a line-by-line dialogue with the user.

PROMPT CHARACTERS

When a connection is first established between a terminal and the computer, a message is sent to the terminal requesting the user to log on. As soon as the user has logged on, TEL types a prompt character at the left margin of the next line to indicate that requests may be entered. Thereafter, a prompt character is sent to the terminal following a completed request, an error, or an interruption by the user. If the services of a subsystem are requested, the subsystem identifies itself with a different prompt character.

The prompt characters used by TEL and all subsystems that carry on line-by-line, rather than intraline, dialogues with the user are as follows:

TEL	!
FORT4	>
META	>
BASIC	>
EDIT	*
FDP	@
DELTA	bell
PCL	<
LINK	:
SUPER	-
CONTROL	-
Libraries	?

ECHOING CHARACTERS - ESC E

For terminals operated in echoplex (full-duplex) mode, character echoing – display on the terminal's output device of characters typed in – by the system is normally 'on', but can be turned off, and on again, etc., at the user's discretion (e.g., to suppress printing of passwords or other security-related information). Successive uses of the ESC-E key sequence toggles the echoplexing on/off state. For terminal units operated in local-printing (half-duplex) mode, characters typed at the terminal are automatically printed by the terminal. When operating in local-printing mode, the user will need to turn echoplexing off to avoid redundant echoing by the system. (In half-duplex, a direct electrical connection exists between the keyboard and printer, via the modem unit.)

ERASING CHARACTERS - RUBOUT OR ESC RUBOUT

Depressing the RUBOUT key (or the ESC RUBOUT sequence) erases the last unerased character. UTS responds by typing

a backslash (\) to indicate it has effectively backspaced and erased. On terminals that can backspace, backspacing does not erase. Thus, it is possible to overstrike characters as well as to erase them.

ERASING THE CURRENT INPUT LINE - ESC X

The current input message (one line or less) is erased by depressing two keys ESC and X sequentially. UTS types a left-facing arrow or underscore, returns the carriage to the position at the beginning of input on the next line, and returns control to the user without further comment. The correct message may then be entered.

CANCELLING ALL INPUT AND OUTPUT - CONTROL X

Depressing the CONTROL and X keys simultaneously will cause all input (including messages typed ahead) and all output to be deleted. If an input operation was pending, additional action is identical to that for ESC X above.

ENTERING BLANK LINES

Blank lines are usually ignored by subsystems. However, some subsystems, in certain modes, treat a blank line as a command to change to a different program control level.

RETYPING THE CURRENT LINE - ESC R

When the ESC R sequence is received, the carriage is returned to the position at the beginning of input on the next line and all characters accumulated will be retyped by the system. The user is then allowed to complete the message.

ENTERING MULTILINE RECORDS – LOC RET, ESC LINE FEED OR ESC RET

On a terminal unit having an inherent line-width limit of less than 140 (e.g., Teletype Models 33, 35, and 37), a single, multiline record may be entered in either of two ways:

1. Using the local carriage return key marked LOC RET, if present, to "break" the input line without releasing it to the system.

2. Using the simulated local carriage return sequence ESC RET or ESC LINE FEED for the same purpose.

TERMINATING LINES

When TEL or a subsystem that carries on a line-by-line dialogue is in use, an "end-of-message" is signaled by depressing either the RETURN or LINE FEED key. Depressing the CONTROL and L keys simultaneously signals an end-of-message" and an "end-of-page". FS, RS, US, and GS (see Table A-3) signal "end-of-message" without carriage motion or character printing. ESC and then F signals "end-of-file". Each read operation at the terminal specifies a maximum number of characters to be read (never more than 140). If this number is reached, "end-of-message" is signaled.

TYPING AHEAD

COC routines allow 'type-ahead' operations. Key strokes (or paper tape frames) that are input by the user before the system requires them will be retained until an M:READ is issued.

PAGINATION AND LINEATION - CONTROL L

Pagination and lineation are controlled by UTS so as to provide 8-1/2 by 11-inch pages with 1-inch margins at the top and bottom of each "page". This assumes a 9-1/2 inch platen, giving 85 characters to the line on XDS 7015; Teletypes provide for 72 characters. UTS counts lines to give 54 lines per page. Pagination can be requested directly by depressing the CONTROL and L keys simultaneously. Pagination consists of the following:

1. Blank lines to the page bottom.

2. A heading line containing date, time, user identification, terminal identification, and page number.

3. Five additional blank lines.

4. User heading line, if any.

Other settings of platen width and page length may be made with a TEL command.

SIMULATING TAB STOPS - ESC T

The user can enter tabulation characters into his terminal input, either with the CONTROL and I key combination or ESC I sequence on teletypewriter units, or the TAB key on terminal units that have it. UTS simulates tabbing by typing (echoing) successive blank characters. Tab-stop values for this simulation can be set or changed by the TABS command. This tab simulation is under the user's control: it is normally 'on' at the beginning of a terminal session, but the user can turn it off, and back on again, with the ESC-T key sequence (i.e., successive uses of the ESC-T sequence has a toggle-switch effect on tab simulation each use reversing the previous on or off state). With tab simulation on, any tab characters

either sent from the terminal or received for transmission to the terminal are replaced at the terminal by an appropriate string of blanks (in lieu of mechanical tabulation). If no tab-stop values are set, each tab character is replaced by a single blank. The state of tab simulation does not determine whether or not blanks are substituted for a tab character in the input stream received by the processor or program requesting the input.

SIMULATING TAB CHARACTERS - ESC I

The ESC I sequence is treated exactly as a tab character. This function is provided for terminals that are not equipped with a TAB key.

INSERTING SPACES - ESC S

One or more spaces are normally inserted into the terminal input stream in place of a tab character (i.e., the tab characters themselves are not normally passed to the processor or program requesting the input). When space-insertion mode is on (initial state), each tab character is replaced in the input stream by an appropriate number of blanks if tab settings are in effect. If there are no tab settings, only a single space is inserted. This space insertion is under the user's control, however: he can turn space-insertion mode off by use of the ESC-S key sequence, causing the tab characters to remain in the input stream. (This can result in a significant space saving in large files.) Successive uses of ESC-S toggles the space-insertion mode from on to off, off to on, etc.

SETTING THE TAB RELATIVE MODE - ESC C

Normally tabs are considered to be physical carriage positions. If the tab relative mode is active, tabs on input, are considered to be offset from the position of the carriage at the beginning of input.

The tab relative mode is toggled on or off by the ESC C character sequence. The tab relative mode is initially set to OFF.

RESTRICTING INPUT TO UPPER CASE - ESC U

When the character pair ESC U is received, a flag that controls alphabetic characters is toggled. When set, all lower case letters received are translated to their upper case counterparts.

INTERPRETING UPPER CASE AS LOWER CASE - ESC)

Receipt of the ESC) sequence causes all subsequent upper case alphabetic characters to be interpreted as the corresponding lower case character until receipt of the ESC (sequence. The parenthesis are echoed, thus bracketing the characters that were interpreted as upper case. This feature is provided to enable terminals that are upper case only to input lower case characters.

EXITING THE LOWER CASE INTERPRET MODE - ESC (

The ESC (character sequence removes the effect of the ESC) sequence (described previously).

TYPING COMMANDS

Except for a few declaratives, commands take the form of imperative sentences. They consist of an imperative verb followed by a direct object or list of objects. Indirect objects usually follow a preposition but may follow the verb with elision of the implied direct objects. Minor variations of this structure are expressed as encoded parentheticals following either the verb or one of the objects. Individual elements of a list of objects are set off from one another by commas.

Common rules of composition are applicable to commands. Words of the language, numerals, object identifiers, and other textual entities may not be broken by spaces. Otherwise, spaces may be used freely. For purposes of scanning commands, both by machine and by human eye, this rule has a simple interpretation. Leading spaces are skipped over in a left-to-right scan for the next syntactic element of a command, and trailing spaces are treated as terminators for words, numerals, and other textual entities. In terms of machine scanning, tabs are treated as spaces.

Since it is impossible to determine whether or not trailing characters in a command are in error, a unique code that identifies the end of the command is recognized as a syntactic element. For TEL and the subsystems that carry on a line-by-line dialogue, this is either a LINE FEED or CARRIAGE RETURN code.

DETECTING AND REPORTING ERRORS

The primary object of the UTS error detection procedure is that user information should not be destroyed by an attempt to execute a command that cannot be carried through to completion. To ensure that each command is at least formally valid, TEL and all subsystems that carry on a line-by-line dialogue always parse an entire command before starting an operation.

Error messages sent to a terminal are as terse as possible since the majority of errors are easily found once the fact that an error exists has been brought to the attention of the user.

The error messages and actions initiated by the errors are contained at the end of each chapter for the system or subsystem to which they apply. Many subsystems use the following format for reporting garbled, malformed, or unintelligible commands:

EH? @ n

where n gives the character position where the confusion was first encountered.

INTERRUPTING UTS

UTS can be interrupted whenever it, one of its subsystems, or a user program has control of the keyboard. Subsequent control depends on which interrupt keys are used and which subsystem or user program is in control.

CONTROL Y, ESC Y, OR ESC ESC

Regardless of what program is in control of the keyboard, the operation can always be interrupted by simultaneously depressing the CONTROL and Y keys (or by typing the ESC Y sequence or the ESC ESC sequence.) UTS responds by stopping the current operation as soon as there is a convenient breakpoint and turning control over to TEL. All input received prior to this key-in that has not yet been read by the program will be erased.

BREAK

If UTS, one of its subsystems, or a program explicitly requesting break control is in control of the keyboard, the operation can be interrupted by depressing the BREAK key. This action gives control to the program that is currently in communication with the terminal.

A succession of four or more BREAK signals always returns control directly to TEL. There are two reasons for this return to TEL. First, some actions can only be stopped at points of convenience and others have so much inertia they cannot be stopped at all. Second, machine or program errors may have disabled the program's response to the BREAK signal. However, it must be emphasized that depressing the BREAK key one time does not constitute a preemptive request for the services of TEL as does depressing the CONTROL and Y keys (or the ESC Y or the ESC ESC sequence).

The precise handling of interruptions by subsystems is defined by the subsystems. The handling of interrupts by object programs is defined by the calls these programs can make on UTS services. If the user does not have break control, interruption of an object program always causes a return to TEL. In general, interruption of the system or any of its subsystems results in termination of the current operation as soon as possible and return of control to the user after the appropriate prompt character has been typed.

ESC Q

Teletype users may request acknowledgment from the system at any time by use of the ESC Q sequence. The system will respond by sending two exclamation points (!!) to the terminal. No other action is taken by the Monitor.

PAPER TAPE INPUT

Paper tape may be punched off-line on Teletype terminals and subsequently read on-line after the user has logged on and a prompt for data on the tape has been issued. The same characters that are keyed in during on-line input may be punched into paper tape. The procedure for reading paper tape on-line is as follows:

1. Insert the paper tape in the paper tape reader.
2. Depress the X-ON (Q^C) key. This will start the reading of the tape by the paper tape reader under control of the computer.
3. Depress the X-OFF (S^C) key to turn off the paper tape mode (read operation).

Rubout characters are ignored during a paper tape read operation. This enables the user to use rubout characters to delete unwanted characters as in normal paper tape operation.

The paper tape read mode is set when a DC1 character (X-ON) is received from the Teletype. It is reset (to normal processing) when a DCT3 character (X-OFF) is received. Characters that are input through the keyboard while the Teletype is in the paper tape mode are normally received after the reader reaches the end of the tape or the tape is removed from the reader.

Restrictions:

1. Line feed (LF) characters received after any other activation condition is reached are ignored unless Delta is reading.
2. The full duplex paper tape facility requires the X-ON, X-OFF option on the Teletype.

HALF DUPLEX PAPER TAPE READING MODE

A special mode is available for half duplex terminals that are reading paper tape. While in this mode, no attempt is made by the Monitor to turn the tape reader off or on. Input is accepted until available buffer space is exhausted. No program output, prompt characters, or echoes are sent to the terminal because the mode renders the terminal incapable of accepting output.

The half duplex paper tape mode is entered upon receipt of an ESC P sequence or upon receipt of an X-ON character while in the non-echoplex mode (controlled by ESC E). The mode is exited by a balancing ESC P sequence or by an X-OFF character if it was initiated by X-ON.

2741 AND TELETYPE DIFFERENCES

In addition to the differing code sets that are translated in a straightforward way, certain unique features of 2741 terminals must be treated in a special way. First, use of the 2741 terminal is proprietary. Both computer and user must, in turn, explicitly release control of the typewriter to the other. Second, two code sets and two keyboard arrangements

for the 2741 EBCD and Selectric® terminals are supported and must be properly identified at log-on time. Third, the important functions provided on Teletypes by the ESC and BREAK keys are combined in the 2741 ATTN key. Finally, a line editing mode that uses backward and forward spacing to position the carrier for character replacement by overstriking is introduced.

LINE STATE

Unlike Teletypes, 2741 terminals cannot transmit and receive at the same time. The 2741 operator can type only when the computer has unlocked the keyboard. The computer can type only when the operator has locked the keyboard by ending his message with a carriage return or attention character.

LOG-ON

When a Teletype line is connected to UTS, a log-on message is automatically sent to the terminal. Logging on from 2741 lines must be handled differently since the keyboard is initiated for user input when the line is connected and the code set and keyboard arrangement are unknown to the computer at this point. The user at the 2741 terminal must identify the code set and type of keyboard before logging on by sending an asterisk (*) followed by a carriage return character. From this point on, the standard log-on sequence is followed.

BREAK AND ESC

Separate BREAK and ESC keys are not present on 2741 keyboards. On these keyboards, the BREAK and ESC functions are performed by the ATTN key. During input, while the keyboard is unlocked, depressing the ATTN key sends an EOT character to the computer. During output, while the keyboard is locked, depressing the ATTN key sends a break signal to the computer. When an EOT character is input to the computer, an escape sequence is performed. The control function is represented by the character input just prior to the EOT. If EOT is received when no other input is received, it is interpreted as a break.

COC ROUTINE

A number of functions are performed in the COC routine to accommodate 2741 terminals. These functions are outlined below.

LOG-ON PROCEDURE

The proper translation table is determined by a special dial-up procedure for 2741 lines. When the asterisk key is depressed, a different code is transmitted for the EBCD and Selectric code sets (both APL and standard versions). This character (*), followed by a carriage return character, is the protocol for 2741 lines to log on with the proper translation table. If the asterisk character is not entered just prior to the carriage return, a space and backspace are transmitted to indicate that the line has been connected but the translation table has not been determined. The procedure can then be repeated.

SPECIAL CHARACTERS

Backspace: The normal mode for processing backspace characters is to put the character in the user's buffer and to include it in the count of characters received. See below for backspace-overstrike editing.

Tab: Tab characters are processed like Teletype except that on input spaces are not echoed to the terminal to position the carrier to the next tab stop since actual physical tabs are available on the 2741 keyboard.

Attention: The ATTN key performs the BREAK and ESC functions of a Teletype terminal (see the section titled "BREAK and ESC"). When a character that represents a control function is detected by the COC handler, a backspace and underscore are transmitted to the terminal to identify the character as part of an escape sequence. An exception is the escape sequence used to retype a line (R ATTN) which results in an R being typed at the terminal before the carrier is returned and the line is retyped.

Lowercase Carriage Return (Carriage Return): The input message is terminated and a carriage return character (X'0D') is put in the user's buffer.

Uppercase Carriage Return (Line Feed): The input message is terminated and a line feed character (X'15') is put in the user's buffer.

CONTROL CHARACTERS

Certain terminal characters perform control functions if preceded by an ESC on the Teletype or if followed by an ATTN (EOT) on the 2741. For the control characters listed below, the function performed is the same on both types of terminals:

- C - Tab relative mode.
- F - End of file.
- L - Form feed.
- Q - Acknowledge.
- R - Retype.
- S - Toggle space insertion.
- T - Tab.
- U - Restrict code to upper case.
- X - Cancel Line.
- Y - Escape to Monitor.
- (- Upper case shift.
-) - Lower case shift.

The function of the following 2741 characters do not correspond directly to Teletype characters.

® Registered trademark of the IBM Corporation.

BACKSPACE: A BACKSPACE ATTN sequence on the 2741 is the same as the Teletype ESC RUBOUT sequence, except in the backspace edit mode (which is discussed below.)

T: The tab simulation mode is switched from on to off or vice versa with the T ATTN sequence. If on during output, enough blanks are sent to the terminal to move the carrier to the next higher tab position. Since actual physical tabs are available on 2741 keyboards, the tab-simulation mode state is ignored during input.

B: Since a break signal is sent to the computer only if the keyboard is locked, the B ATTN sequence simulates a break when the keyboard is unlocked and is treated like a break signal on a Teletype during input. If no other input has been received, ATTN by itself is interpreted as a break.

SPACE: The input line is terminated and the end-of-message character US (X'1F') is put in the user's buffer when the SPACE ATTN sequence is received.

O: The backspace edit mode is switched from off to on or vice versa when the O ATTN sequence is received. If on, backspace editing is performed inside the COC handler (see below).

BACKSPACE EDITING

The standard mode for processing backspace characters is to pass the character to the user's buffer and to include it in the count of characters received. The backspace edit mode is invoked when the character sequence O ATTN is received. If a backspace is received in this mode, the pointer into the input buffer for the next character is saved and then decremented by one character. Characters are processed in the following manner until this pointer is incremented to its original position:

1. Additional backspace characters decrement the pointer that points to the current character in the input buffer by one.

2. A space increments the pointer that points to the current character in the input buffer by one.

3. The character sequence BACKSPACE ATTN is an explicit blank. The current character in the buffer is replaced with a blank and the pointer that points to the current character in the input buffer is incremented by one. Two SPACE characters are sent to the terminal to correct the carriage position.

4. End of message characters (NL, L ATTN, F ATTN, or SPACE ATTN) cause the message to be terminated and the appropriate end of message characters to be placed at the end of the current line (after the rightmost character).

5. Any character other than another backspace, blank, rubout, or end of message overlays a character in the buffer; the pointer that points to the current character in the input buffer is incremented by one.

When the pointer that points to the current character in the input buffer becomes equal to its original value, normal processing of input characters resumes.

SUMMARY OF 2741 AND TELETYPE DIFFERENCES

Table 2 summarizes the differences between 2741 terminals and Teletype terminals. (Refer to Table A-5 for substitutions for characters nonexistent on 2741 terminals.)

Table 2. Summary of Differences Between 2741 and Teletype Services

Function	Teletype	2741
Get log-on message	BREAK	* and CRLF if dialing up. ATTN if line is already connected.
Erase line	ESC X	none
Tab relative	ESC C	C ATTN
Suppress lowercase	ESC U	U ATTN
Uppercase shift	ESC (	(ATTN
Lowercase shift	ESC)	) ATTN
Erase last character	RUBOUT	BACKSPACE ATTN
Tab	ESC I	TAB
End of input	FS, RS, US, GS (L^{CS}, N^{CS}, O^{CS}, M^{CS})	SPACE ATTN
Line continuation	ESC CR	N ATTN
Retype	ESC R	R ATTN
Toggle tab simulation mode	ESC T	T ATTN
Toggle space insertion mode	ESC S	S ATTN
End of file	ESC F	F ATTN
Monitor escape (to TEL)	ESC ESC, ESC Y, or 4 BREAKs	Four ATTNs. Also, Y ATTN if input.
Break	BREAK	B ATTN on input or ATTN on output.
Toggle backspace edit mode	None	O ATTN
Form feed	ESC L	L ATTN
Half duplex paper tape	ESC P	none
Toggle ECHO mode	ESC E	none
Acknowledge	ESC Q	none
Erase all input and output	CONTROL X	X ATTN

3. TERMINAL EXECUTIVE LANGUAGE

INTRODUCTION

The Terminal Executive Language (TEL) is the principal terminal language for UTS. Most activities associated with FORTRAN and assembly language programming can be carried out directly in TEL. These activities include:

Major Operations

Composing program and data files.
Assembling and compiling programs.
Linking object programs.
Loading programs and initiating execution.
Initiating debugging operations.
Managing and backing up files.
Submitting batch jobs.
Calling subsystems.
Interrupting, continuing, and terminating execution.

Minor Operations

Checkpointing on-line sessions.
Assigning I/O devices and DCB parameters.
Determining on-line user status.
Listing system load parameters.
Setting simulated tab stops.
Changing terminal type.
Changing terminal platen size.
Sending messages to operator.
Printing or punching output.

MAJOR OPERATIONS

Figure 2 illustrates the sequence in which major operations normally take place. Capitalized words identify TEL commands and UTS subsystems that are used to carry out the various programming activities.

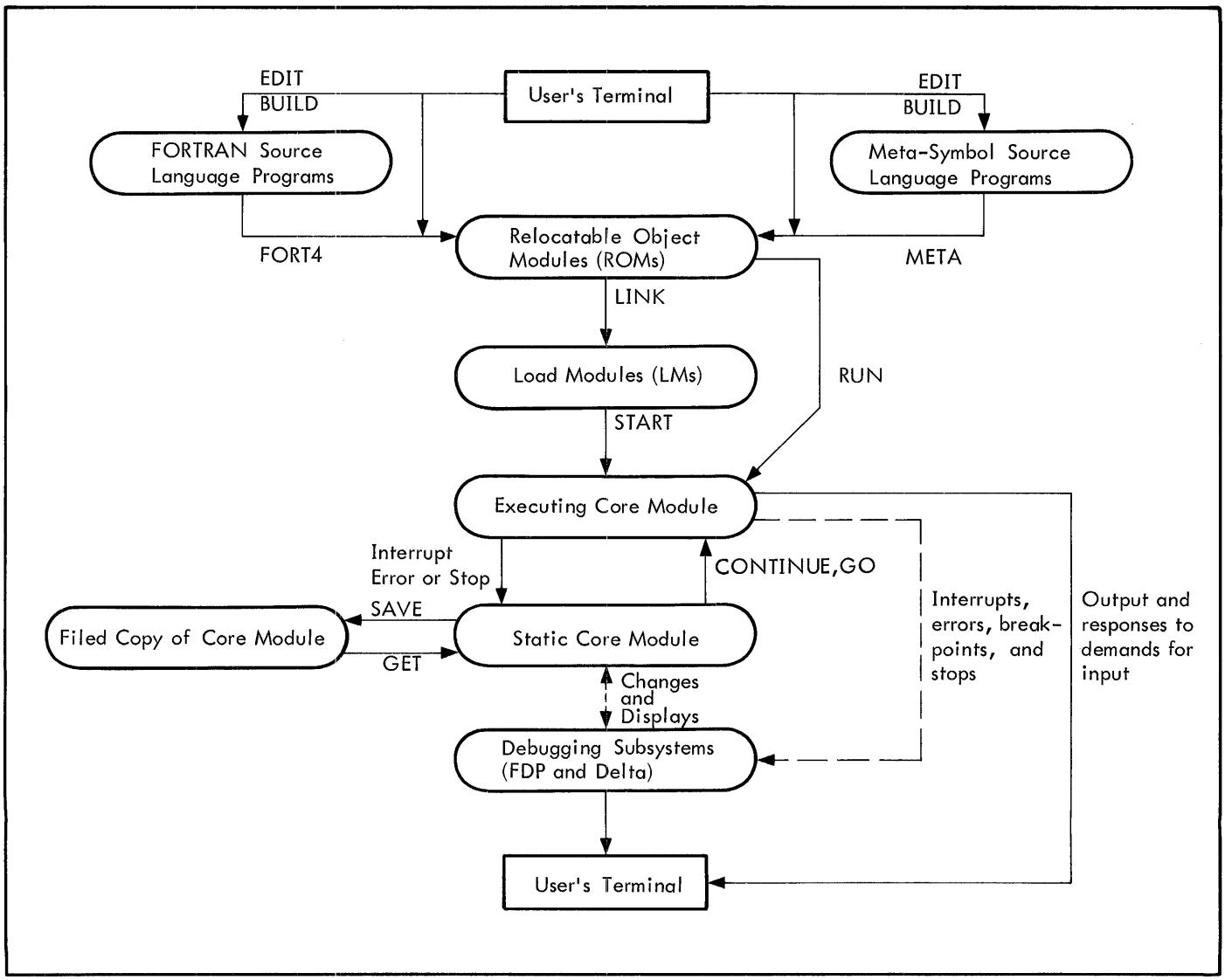


Figure 2. FORTRAN and Assembly-Language Programming

A Meta-Symbol or FORTRAN program may be composed on-line in one of two ways. It may be composed and filed away by the Edit subsystem, which is called by the EDIT or BUILD commands, or entered directly from the terminal one line at a time after Meta-Symbol or FORTRAN has been called with a META or FORT4 command. In both cases, program assembly or compilation is initiated by the META or FORT4 command and a relocatable object module (ROM) and program listing may be produced. Output is directed to the files or devices specified by the user.

Relocatable object modules that have been assembled or compiled separately are put together by the LINK command to form a load module (LM). On completion of the linking operation, execution is started by the START command. Or, if desired, both linking and execution can be initiated by a single RUN command.

Debugging activities are initiated by starting the execution of a load module under control of one of the debugging subsystems, Delta or FDP. Delta is most appropriately used for debugging Meta-Symbol programs but may be used for debugging any program. It may always be called into association with an executing user program for aid even after execution has begun. FDP is used for debugging FORTRAN programs.

An executing program becomes a static core module whenever it is interrupted or whenever an error occurs. This static core module can be stored by the SAVE command and retrieved later by the GET command; it can then be restarted with the CONTINUE or GO command.

COMPOSING PROGRAM AND DATA FILES

The Edit subsystem provides for line-at-a-time composition and editing of files and is called in either of two ways:

EDIT fid

BUILD fid

File identification (fid) has the following format:[†]

name [.account | .account.password | ..password]

where

name is the name of the file and may have a maximum of 31 characters. (TEL, Link, and Load allow a maximum of 10 characters.)

account is the account number of the file and may have a maximum of eight characters. A user may not create (BUILD) a file in any account other than the one under which he is running. He may not EDIT a file in another account unless he has write access to the file.

password is the password for the file and may have a maximum of eight characters.

Account and password are optional, defaulting to the log-on account and no password if omitted. The minimum character set allowed in the three elements of a fid for Edit and all on-line systems is as follows:

A-Z a-z 0-9 _ $ * % : # @ - backspace

When called by EDIT, the Edit subsystem opens the specified file (fid) for updating, issues the Edit prompt character (*), and then waits for input of commands. The commands of the Edit subsystem may then be used to update the file. (Reference: Chapter 6.)

When called by BUILD, the Edit subsystem assumes that a new file is to be entered a line at a time, beginning with line number 1.000 and continuing in increments of one. Edit responds by printing the number of each line at the left margin and waiting for entry of the line. The end of the file is signaled by entering an empty line. Edit is available for corrections and other editing operations after the file has been keyed in.

ASSEMBLING OR COMPILING PROGRAMS

TEL has two commands that permit a program to be assembled or compiled into a single ROM. These commands are

META [sp] [ON | OVER [rom] [,list]]

FORT4 [sp] [ON | OVER [rom] [,list]]

where

sp specifies a source program and may be either a file identification (fid) or the terminal identification (ME). If no source file is specified, TEL assumes input is from the terminal (ME). (sp is assigned to the M:SI DCB.)

ON indicates that ROM output is to be on a new file.

OVER indicates that ROM output is to be over an existing file.

rom specifies that the relocatable object module produced by assembly or compilation is to be directed to a specific file (fid). If no ROM is specified, output is directed to a special file that may subsequently be referenced by a dollar sign. (rom is assigned to the M:GO DCB.)

list specifies that listing output is to go to a file (fid), a line printer (LP), or the terminal (ME). If list is not specified, no listing output is produced. (list is assigned to the M:LO DCB.)

Whenever TEL encounters an input specification (sp) designating the terminal (ME), the program to be assembled or compiled must be entered through the terminal a line at a time. The end of program input is signaled by an end-of-file that is produced at the terminal by the key sequence ESC and F.

Any assignments made at a job step within META and FORT4 commands apply to all subsequent job steps, except for

[†] This definition of file identification is intended whenever fid is used in a command specification; PCL commands allow a larger character set, however.

source input which always reverts to the terminal. These assignments may be changed by subsequent assignments either by META and FORT4 or by the OUTPUT, LIST, and COMMENT commands described below.

CONTROLLING OUTPUTS

Control over output may be exercised before the FORT4 and META commands are entered. This is accomplished by the commands

OUTPUT [ON / OVER rom]

LIST [ON / OVER list]

COMMENT [ON / OVER list]

OUTPUT specifies the destination of ROM output and may designate a file (fid) only. LIST specifies the destination of listing output; COMMENT specifies the destination of error commentary. LIST and COMMENT may designate a RAD storage file (fid), a line printer (LP), or the terminal (ME).

Output parameters set up in this way are valid across job steps from the time given until the session is terminated or until reset. They may be reset by other LIST, OUTPUT, COMMENT, and SET commands or by META and FORT4 commands that specify output.

Whenever output parameters are specified by the LIST, OUTPUT, and COMMENT commands, execution of multiple META or FORT4 commands without output parameters will continue to place the output from these operations on the same files. This is accomplished by file extension (see "Extension of Output Files").

Examples:

1. Assume a FORTRAN source program, on file A, is to be compiled. The name of the rom file is C, the name of the list file is D. Both C and D are new files. All files have the user's log-on account and password.

```
!FORT4 A ON C,D (RET)
!
```

2. Assume the same conditions as in the previous example except that the output files are to be specified by LIST and OUTPUT commands. Error commentary is to go to the terminal.

```
!OUTPUT ON C (RET)
!LIST ON D (RET)
!COMMENT ON ME (RET)
      :
      :
!FORT4 A (RET)
```

Output from an assembly can be interrupted and turned off at any time by one of the following commands:

```
DONT LIST
DONT OUTPUT
DONT COMMENT
```

and turned on again subsequently with LIST, OUTPUT, or COMMENT, respectively.

Output need not be directed to the terminal to be controlled by these commands. Error commentary is normally directed to the terminal and accompanies listing output, if specified.

After output has been redirected, as desired, processing is continued by the CONTINUE command. If continuation is not desired, processing may be discontinued by the QUIT command.

Example:

Assume that source file B is to be assembled. ROM output is going to C and listing output is going to the terminal. After part of the output has been typed, the output operation is interrupted and the listing is discontinued.

```
!META B ON C, ME (RET)
(Listing output.)
(User presses BREAK key.)
!DONT LIST (RET)
!CONTINUE (RET)
(Assembly is completed with no more listing.)
!
```

EXTENSION OF OUTPUT FILES

File extension is a convention by which records are added to an output file by successive job steps. Each time the file is opened, the file pointer (RAD, disk pack, labeled magnetic tape, etc.) is positioned to a point immediately following the last record in the file. Thus, when additional output is produced it is added to the previous contents of the file, thereby extending it. File extension simulates output to physical devices, such as line printers or typewriters, when output is actually directed to a file.

File extension takes effect at the time UTS opens system output DCBs. The output DCB's that are affected by file extension are those that are normally assigned by default to devices, either in batch or on-line operation, but that are explicitly assigned to a file (e.g., on RAD storage) at the time the DCB is opened. These DCB's include M:AL, BO, CO, EO, LL, LO, PO, SL, and SO. The M:GO DCB is also subject to file extension.

File extension is discontinued when a file is reassigned with a SET or other output-controlling (e.g., OUTPUT) command, or when a file is opened with an OPEN procedure call that specifies an explicit file name In these cases, a new file is created. Extension of the GO file is terminated following a LINK or RUN command.

ERROR HANDLING AND END ACTIONS

Whenever an operation is aborted, either because the operation cannot be continued or because a QUIT command is issued, UTS restores certain specifications before reporting and returning control to the user. In particular, aborts occurring outside of TEL (within compilers, assemblers, or user programs) result in all previous output specifications and file assignments being restored to the specifications in effect at the beginning of the job step.

When syntax errors are encountered in input messages, the input is erased and an error message is sent to the terminal. An entirely new command must be issued.

ENTERING PROGRAMS FROM TERMINAL

Whenever the input designator ME is encountered, such as in the processing of META or FORT4 commands, the carrier is returned to the left margin of the next line and a prompt character is sent to the terminal. A program statement can then be entered. It is followed by a carriage return or line feed character to identify the end of the statement. Error commentary, if any, is sent to the terminal immediately thereafter. The end of source input is signaled either by the ESC and F keys (for META and FORT4) or by the appropriate subsystem command (such as END).

To aid in formatting, print columns on the terminal's platen are in a one-to-one correspondence with card columns. Trailing blanks are assumed for short lines. The terminal's tab stops should be set by the user to conform to the programming language being used and will be simulated if tab simulation is in effect. For FORTRAN, a single tab stop is set at column 7. For Meta-Symbol, tab stops are set at columns 10, 19, and 37.

The handling and simulation of tab stops is described in Chapters 2 and 10. Briefly, tab simulation works in the following way. Spaces are sent to the terminal to bring the carrier to the position indicated by the next tab position that has been set. Tabbing requested when the carrier is beyond the last set tab position is simulated by a single space.

DEBUGGING INFORMATION

The ROM output of a Meta-Symbol assembly contains sufficient information for subsequent debugging at assembly-language level under Delta. However, for symbolic debugging, a symbol table is needed. The user can get a symbol table by using the SD option during assembly.

To debug FORTRAN programs under FDP, additional information must accompany the compiled code. This information is not normally produced by the compiler since it increases the size of object programs and decreases their execution speed. To produce the information for a specific compilation, the DEBUG option for the FORT4 command must be used (Chapter 4).

LINKING OBJECT PROGRAMS

ROMs and LMs are both representations of programs and data. ROMs are designed so that they can be efficiently combined with other ROMs, and LMs are designed so that they can be efficiently translated into executable programs and loaded into core. Both may be pictured as bodies of potential machine code to which symbol tables are appended. These symbol tables list the correspondence between the symbolic identifiers used in the original source program and the values of virtual core locations that have been assigned to them. Some of these identifiers are defined and referenced within the same module and are internal symbols. Others are defined (DEF) and referenced (REF) in separate modules and are global symbols.

Functionally, these modules can be compared with black boxes with labeled connectors dangling from them, some pointing out and others in. The labeled connectors are the global symbols associated with the modules; the internal connections have all been sealed and are hidden. In the process of linking modules, internal symbols associated with the constituent parts of the new load module are sealed and hidden, but all global symbols are still visible.

Continuing the black box analogy, if a module is split open, a jumble of internal connections should be visible. If the module has been tested and is ready for production, the internal connections need not be labeled. However, if the module is still in the debugging stage, the labels may be necessary. An option is provided in the LINK command to indicate that the internal symbols associated with a module are to be kept with the resulting load module.

Note that LINK is a one-pass loader. It is beyond the scope of a one-pass loader to handle the multiple use of dummy sections with code that involves REFs that have not been satisfied.

SIMPLE LINKAGES

Most commonplace linkages of ROMs can be carried out directly in TEL and are initiated with the LINK command. The format of this command is

LINK rom[,rom]...[,rom] [ON / OVER lmn]

where

rom specifies a relocatable object module and may be either a file identification (fid) or a dollar sign. The dollar sign designates the most recent compilation or assembly.

lmn (load module name) specifies where the load module is to be placed and may be a file identification (fid) or dollar sign. If lmn is omitted, the resulting load module is placed in a special file and is available for subsequent execution (see Initiating Execution).

Example:

Assume that a load module, E, is to be created for execution from files A, B, C, and D. Public library P1 is to be associated with the load module to satisfy external references. No search of the system library is required. (See the section titled "Searching Libraries".)

```
! LINK (NL) A,B,C,D ON E (RET)
!
```

LOAD MODULE SYMBOL TABLES

A load module consists of three parts: a body of code, a table of global symbols, and a table or set of tables of internal symbols. Each table of internal symbols is associated with a specific input module (ROM) and is identified by the file name of that module. This identification is used by Delta to specify the set of internal symbols to be used for debugging. The section titled "Merging Internal Symbol Tables" describes what happens to the tables when a ROM is linked with other ROMs.

An optional parameter is used with the LINK command to indicate when the internal symbols of an input module are to be kept with the resulting load module. The rules governing this parameter are as follows:

1. The parenthesized letters "NI" preceding the file identification specify that internal symbols for that module are not to be included in the load module; the parenthesized letter "I" specifies that internal symbols are to be included.

2. Once given, a specification applies to all subsequent modules in the command until the occurrence of a new specification.

3. In the absence of any specifications, all internal symbols are retained.

Example:

Assume that a load module, E, is to be created from files A, B, C, and D. Public library P1 is to be associated with the load module to satisfy external references but no search of the system library is required. Internal symbol tables are to be created for files A and D but not for files B and C.

```
!LINK A, (NI)B, C, (I)D ON E ↵
!
```

MERGING INTERNAL SYMBOL TABLES

Keeping the internal symbol table for each input module uniquely identified in a load module is useful when duplicate names have been used in the programming of the input modules. However, if duplicate names have not been used, several symbol tables may be merged into a single table in the resulting load module by enclosing the list of input modules named in the command in parentheses.

LINK (rom[,rom]... [,rom]) [ON / OVER lmn]

Only one level of parentheses is allowed. Multiple uses of internal identifiers are resolved by assigning them to the object they identify in the first (reading from left to right) input module with which they were associated. The identification given to the internal symbol table is the name of the last input module specified in the merge.

Example:

Assume that a load module, E, is to be created from ROMs A, B, C, and D. The internal symbols for files D and A are to be merged. The internal symbols for B and C are not to be included in load module E.

```
!LINK (A,D), (NI) B,C ON E ↵
!
```

SEARCHING LIBRARIES

Unsatisfied external references are resolved by specifying the order and identification (lid) of libraries to be searched after the input modules have been linked. A list of library identifications, (lid) separated by commas, is appended to the list of modules in the LINK command and is separated from the module list by a semicolon.

LINK rom [,rom]... [,rom] [ON / OVER lmn] [;lid [,lid] ... [,lid]]

where lid specifies a library file identification (fid). In the absence of any other specifications, public library P1 is associated with the load module to satisfy external references and the system (ROM) library is searched if necessary. Optional search codes may be entered anywhere in the command except between a preposition and its object. For convenience, they are shown below immediately following the command verb.

LINK [codes]rom[,rom]... [,rom] [ON / OVER lmn] [;lid [,lid]... [,lid]]

where codes may be one or more of the following:

(L) specifies that the system library is to be searched to satisfy external references that have not been satisfied by the program. (This is a default option.)

(NL) specifies that a system library search is not required.

(P_i) specifies that the ith public core library is to be associated with the program to satisfy external references. Only one public library may be associated with a program. P0 and P1 are supplied by XDS; P1 contains a subset of the FORTRAN library subroutines; P0 includes P1 and the FORTRAN Debug Package. Additional public libraries must be named P2-P9 and J1-J9. (P1 is a default option.)

(FDP) equivalent to (P0).

(NP) specifies that a public core library is not required.

The sequence of the library search is as follows: User libraries are searched first, the public library is associated, and the system library is searched.

Examples:

1. Assume that a load module, E, is to be created from files A, B, C, and D. Internal symbols for files B and C are not to be included in the load module; internal symbols for files D and A are to be merged. Two user libraries, F and G, are to be searched to satisfy external references. Public library P1 is to be associated with the load module but no search of the system library is required.

```
! LINK (D,A), (NI)B, C ON E;F,G ⏎
!
```

2. Assume the same problem as in the previous example except that the system library is to be searched for external references and public library P2 is to be associated with the load module.

```
! LINK (L)(P2)(D,A), (NI)B,C ON E;F,G ⏎
!
```

3. Assume the same conditions as in the first example except that no libraries are to be searched.

```
! LINK (NL)(NP)(D,A),(NI)B,C ON E ⏎
!
```

END ACTIONS AND ERROR DISPLAYS

Options governing error displays consist of parenthesized codes. These codes may be placed anywhere in the command except between a preposition and its object in the same manner as the library search options.

(D) specifies that all unsatisfied internal and external symbols are to be displayed at the completion of the linking process (including library searches, if specified). The unsatisfied symbols are identified as to whether they are internal or external and to which module they belong.

(ND) specifies that the unsatisfied internal and external symbols are not to be displayed.

(C) specifies that all conflicting internal and external symbols are to be displayed. The symbols are displayed with their source (module name) and type (internal or external).

(NC) specifies that the conflicting symbols are not to be displayed.

(M) specifies that the load map is to be displayed upon completion of the linking process. The symbols are displayed by source with type resolution and value.

(NM) specifies that the load map is not to be displayed.

The normal default options are D, C, and NM.

LOADING PROGRAMS AND INITIATING EXECUTION

Any stored load module may be loaded into core and started by presenting TEL with the name of the load module (lmn) as a command verb. Additional parameters may be given to specify assignments. The format of the command is the same as for FORT4 and META commands with a load module name replacing the processor name.

$$\text{lmn [sp]} \left[\begin{matrix}\text{ON} \\ \text{OVER}\end{matrix} \text{[rom] [,list]}\right]$$

where lmn is the load module name and has the following format:

$$\text{name} \left[\text{.[account] [. password]}\right]$$

When lmn is used as a command verb, the default account is interpreted as follows:

name implies the system account.

name. implies the log-on account.

name.account specifies an account and no password.

name.account.password specifies an account and password.

name..password implies the log-on account and specifies a password.

sp is the identification (fid or ME) of the input file to be assigned to the M:SI DCB.

rom is the identification (fid) of the output file to be assigned to the M:GO DCB.

list is the identification (fid, LP, or ME) of the output file to be assigned to the M:LO DCB.

TEL scans the parameters in an attempt to create assignments as it does for the FORT4 and META commands. If the line scan is not desired and there are parameters to scan, the parameters may be enclosed in parentheses since TEL ignores all parameters within the parentheses. Unpaired parentheses are treated as syntax errors.

Examples:

```
! TESTOR ⏎
```

(loads the LM using the system account)

```
! TESTOR. ⏎
```

(loads the LM using the log-on account)

! TESTOR.1234 (RET)

(loads the LM using account 1234)

! TESTOR..SECRET (RET)

(loads the LM using the log-on account and the password "SECRET")

! TESTOR FILEA ON FILEB, FILEC (RET)

(loads the LM using the system account – FILEA is assigned to the M:SI DCB, FILEB to the M:GO DCB, and FILEC to the M:LO DCB)

! TESTOR (ABC(DEF(GHI)JK)) (RET)

(loads the LM using the system account and passes the line image to the program, starting at JIT word location J:CCBUF.)

Two other TEL commands are provided to initiate the execution of a program. One of these commands (START) loads a load module into core and starts execution at its beginning address. The other command (RUN) links relocatable object modules, loads the resulting load module into core, and starts execution.

The format of the START command

START [lmn / $]

where lmn is the name (fid) of the load module to be executed. If lmn is omitted or a dollar sign is specified, the last load module formed by LINK on the $ file is executed.

The RUN command is a combination of LINK and START. It has the following format:

RUN[rom] [,rom]... [,rom]

where rom is the name (fid) of a relocatable object module to be linked, loaded, and executed. A dollar sign may be used for rom to designate the most recent assembly or compilation. If no parameters are given, the result of the last major operation (assembly or compilation) is loaded and executed.

Example:

Assume that file A is to be assembled, loaded, and executed.

! META A (RET)
! RUN (RET)
!

All options of the LINK command may be exercised in the RUN command in exactly the same manner.

Assume there are three modules to be loaded: A, B, and C. The internal symbols for A and B are to be kept with the resulting load module. The internal symbols for C are not. User library D and the system library are to be searched for external references that have not been satisfied by the program. Public library P2 is to be associated with the program.

! RUN (L)(P2)(I)A,B,(NI)C;D (RET)
!

INITIATING DEBUGGING OPERATIONS

Execution of programs can be started under control of either one of two debugging subsystems, Delta or FDP, by appending the word "UNDER" and the name of the debugging subsystem to a RUN command.

RUN... [UNDER DELTA / FDP]

where

DELTA identifies the assembly language debugging subsystem.

FDP identifies the FORTRAN debugging subsystem.

Debugging operations can also be initiated with the START command. This is accomplished as follows:

START...[UNDER DELTA]

After the programs have been loaded into core, control passes to the designated debugging subsystem which sends an identifying message to the terminal and awaits commands.

Delta may also be called when execution has been initiated without it. This is usually done after an interruption by the user or an error comment by the system. In this case, Delta is called by typing

DELTA

FDP may also be initiated by specifying either of the library-search codes P0 or FDP in a RUN or LINK command, e.g.:

RUN (P0)[rom][,rom]... [,rom]

This associates public library P1 and FDP with the user program, thus allowing execution of the program under FDP.

MANAGING AND BACKING UP FILES

File management and information-transfer capabilities are provided by the PCL subsystem (Chapter 5). PCL can be called implicitly, however, at TEL level, via the COPY

command. A very simple form of the COPY command is as follows:

COPY sf{TO[t] / OVER} df

where

sf specifies an input device or a source file on RAD, labeled tape, or disk pack.

df specifies an output device or a destination file on RAD, labeled tape, or disk pack.

The transfer of information to a printer or to the terminal may be aborted by depressing the BREAK key.

The many additional variations of the COPY command, as described in the section on PCL, are also available through TEL.

Files can be deleted with the TEL DELETE command. This command has the following format:

DELETE fid

where fid is the identification of the file to be deleted. The filename in this fid is limited to 10 characters; see also the PCL DELETE. Deletions cannot be interrupted after they have been started.

Files created or modified during an on-line session may be saved by using the BACKUP command. The format of this command is

BACKUP fid

where fid (10-character limit) names the file to be copied to the standard system backup tape. Automatic system restart includes restoration of all entries on the backup tape onto the permanent-file RAD.

The backup process is mechanized by an asynchronous process that handles backups for all users. Therefore, there may be a delay between the time the backup command is issued and the time that the file is placed on tape. Also, by rules of simultaneous file access, the file may be unavailable to the user during the time it is being copied by BACKUP. The user-requested backup process delivers error messages as well as successful completion messages to a keyed file (called MAILBOX) in the user's account. The user may print his MAILBOX file using the command

COPY MAILBOX ON ME

SUBMITTING BATCH JOBS

Programming functions described earlier in this chapter are performed on-line. The user may also compose batch job-control 'decks' on-line, using the Edit subsystem, and submit these to the batch queue for later execution. Optionally, the specification field of the JOB and LIMIT control commands may be left blank and Batch will supply the missing subfields before submitting the job. The command used for this purpose has the form

BATCH fid[,fid,...,fid]

where fid is the identification of a job file to be submitted for batch processing. This command may be executed in batch mode as well as on-line mode.

UTS responds to this command by assigning the batch job a job identification (jid) and sending this message to the terminal or printer (M:LL):

ID = jid SUBMITTED time-date
WAITING: n TO RUN

An automatic job-status report is issued immediately following this message.

The procedure for assigning priorities to remotely submitted batch jobs is the same as the procedure for assigning priorities to jobs submitted at the central site. This procedure is described in the UTS/BP Reference Manual, 90 17 64. But see Table 9, in this chapter, for as-distributed maximum priority for on-line submission and for maximum LIMITS-option values as related to job priority. Batch subsystem error messages are listed in Table 7.

The status of one or more jobs submitted to the batch queue may be interrogated at any time by typing

JOB jid[,jid,...,jid]

where jid is the job identification reported when the job was submitted using the BATCH command. Response is one of the following:

COMPLETE	if the job has been run.
EH?	if the jid is indecipherable.
DOESN'T EXIST	if the job never existed.
RUNNING	if the job is currently in execution.
WAITING: n TO RUN	if the job is waiting to run behind n others.
WAITING TO OUTPUT	if the job has run and symbiont output remains to be printed or punched.

CALLING SUBSYSTEMS

All subsystems are called by typing the subsystem identification. The subsystems respond by identifying themselves and then typing their prompt character at the left margin of the next line before returning control to the user. The

[t] Wherever TO is specified, ON may be substituted.

subsystem identification and prompt character for each subsystem are listed below.

FORT4 >	PCL <
META >	LINK :
BASIC >	
EDIT *	SUPER –
	CONTROL –
FDP @	
DELTA bell	Libraries ?

Example:

Assume that the PCL subprocessor is to be called.

```
! PCL (RET)
PCL B00 HERE
<
```

INTERRUPTING, RESUMING, AND TERMINATING EXECUTION

There are several courses of action that may be taken whenever a major operation, a subsystem operation, or an executing user program has been stopped or interrupted. First, any of the minor operations listed in the next section except the operation of the SET command may be initiated. TEL operations may then be resumed by one of the following commands:

CONTINUE or GO

Second, the operation may be given up completely by entering

QUIT, END, or STOP

In this case, TEL restores certain specifications before returning control to the user (see Error Handling and End Actions).

Third, a new major operation may be initiated. Here the effect is the same as if TEL had been told to QUIT. The sole exception to this occurs when Delta is interrupted during execution of a user program. The program must be initiated again under control of Delta. (Note: When a program is being executed under control of FDP, program-end or the BREAK key is pressed four times. Control goes to FDP if there is an error or the BREAK key is pressed one or two times.)

Three categories of TEL commands may be identified:

1. Commands that may be given any time TEL prompts, even if a program or processor has been interrupted. They are:

BACKUP	JOB
CONTINUE	MESSAGE
DELETE	PASSWORD
DELTA	PLATEN
DISPLAY	PRINT
DONT COMMENT	SAVE
DONT LIST	STATUS
DONT OUTPUT	TABS
GO	TERMINAL

2. Commands that may be given at any time but abort and erase any currently running program or process. They are:

BASIC
BATCH
BUILD
COPY
EDIT
END
FORT4
LINK
META
OFF
PCL
QUIT
RUN
START

3. Commands that result in an error message if given during an interruption of a running program or processor. They are:

COMMENT
GET
LIST
OUTPUT
SET

(And the implicit loading of a program by giving its name.)

MINOR OPERATIONS

Minor operations consist of the operations that support on-line programming. They include checkpointing, assigning I/O devices and DCB parameters, determining current user status, and so on.

CHECKPOINTING ON-LINE SESSIONS

During interruptions of execution, core images of programs may be saved on RAD storage for subsequent recall and continuation. The SAVE command is used for this purpose. The format of this command is

SAVE {ON / OVER} fid

where fid is the identification of the file in which the image should be saved.

A checkpointed core image may be recalled for continuation by the GET command. The format of this command is

GET fid

where fid is the identification of the file to be recalled. This file must be in the user's log-on account. The program may be restarted with a CONTINUE command.

SAVE is implemented in such a way that execution of a program is unaffected by a SAVE-CONTINUE sequence of operations except for the time delay. Especially important is the fact that open files are not closed or repositioned.

A GET operation, however, requires that any current execution be terminated and all files using default close options be closed. This means that current position information is lost for IN and INOUT files (they are effectively rewound) and OUT and OUTIN files are released. Thus, whenever a GET command is issued the user must take responsibility for repositioning of IN files and recreation of OUT files that were open at the time of the save in whatever way is appropriate to continuation of his program.

The collection of I/O assignments made during the job (up to the point of SAVE) and collected in the user's assign-merge table is not preserved, but the active DCBs are. The effect of the current assign-merge activity is therefore carried over to the GET operation through DCBs. The assign-merge table current at GET time has no effect on the retrieved DCBs.

SAVE remembers the names of any shared processors associated with the program that are to be saved. These same named processors are reassociated by the GET command. If the shared processor has changed in the elapsed time between the SAVE and the GET, proper continuation may not be achieved.

Symbiont output that has been produced, say for printer or punch, is packaged for delivery to the appropriate device whenever a SAVE command is given.

ASSIGNING I/O DEVICES AND DCB PARAMETERS

DCB assignments to files or devices, and many DCB parameters, may be set from an on-line terminal. This includes most of the parameters that are set by a batch ASSIGN command and many of the parameters that are set by OPEN and DEVICE procedure calls in a batch program. The command that sets these assignments and parameters is the SET command.

UTS retains all information supplied by SET commands in a permanent table associated with each user. This table is called the assign-merge table and is stored on RAD. At each job step (i.e., each time a new user program or processor is loaded), the information in the assign-merge table is merged into the DCBs associated with the program. An entry for a DCB that is currently in the assign-merge table may be deleted by the command

SET dcb 0

This allows the default assignment (if any) for that DCB to take effect.

Assignments are one of two types: device (printer, punch, magnetic tape, etc.) or file (RAD, disk pack, or labeled magnetic tape). If a DCB that has already been assigned to a device is assigned to a file, the new information replaces the old information in the assign-merge table. The same procedure applies to device assignments for DCBs currently assigned to files. Each DCB assignment requires an entry in the assign-merge table. The total number of DCBs that may be assigned is limited to 12.

Changes to device parameters are added to DCBs assigned to devices. Changes to device parameters for DCBs assigned to files yield an error message.

SET commands may be issued only between job steps, i.e., not during interruptions thereof. Once issued, the information specified by the command for all but the M:SI DCB remains in effect until revoked, regardless of whether one or many job steps are included in the session.

The several formats of the SET command are:

SET dcb 0

SET dcb [oplabel / device / tapecode[tapeid]] [;dopt[;dopt]...[;dopt]]

SET dcb [tapecode[tapeid]/fid / filecode[packid]/fid] [;fopt[;fopt]...[;fopt]]

where

dcb identifies a DCB and is in the form M:x or F:x where x is 1 to 9 characters. (Assignments of M:UC, M:OC, and M:XX are not allowed).

oplabel specifies an operational label (BI, C, CI, etc.). (See Table 3.)

device specifies a device code (CP, PP, LP). (See Table 3.)

tapecode specifies a magnetic tape code (9T, 7T, MT). (See Table 3.)

filecode specifies a secondary storage code (DC or DP). (See Table 3.)

tapeid if followed by /fid, specifies a serial number for a labeled tape and has the form #serial number. The tape is accessed with the serial number applying as both an INSN and an OUTSN. (Serial numbers may contain alphanumeric characters and are 1-4 characters in length.) If not followed by /fid, it may specify an external reel number for free-form tape.

packid must be followed by /fid; specifies a serial number of a private pack and has the form #serial number.

/fid specifies the name of a file on tape or secondary storage. The form is

name [.account / .account.password / ..password]

If not preceded by a tapecode or filecode, filecode DC is implied.

dopt specifies a device option. (See Table 4.)

fopt specifies a file option. (See Table 5.)

Spaces may be arbitrarily used in a SET command between numbers, words, and identifiers but may not be embedded within them.

Examples:

1. Assume that the Monitor DCB for listing output is to be assigned to RAD storage file N under account A with password P.

```
!SET M:LO/N.A.P (RET)
!
```

2. Assume that the Monitor DCB for source input is to be assigned to file M on magnetic tape serial number 4003.

```
!SET M:SI MT#4003/M (RET)
!
```

3. Assume that tab positions 27, 38, 47, and 75 are to be added to the listing output DCB. In addition, the first character of each record of the listing is to control vertical format and the listing is to be double spaced.

```
!SET M:LO;TAB=27,38,47,75;VFC;SPACE=2 (RET)
!
```

If the M:LO DCB is not assigned when the above changes are made, an error message will be sent to the terminal.

DCB ASSIGNMENT CODES

A device assignment is made whenever a SET command contains an expression with an operational label or device code, or a tapecode/tapeid not followed by a file identification. For each assignment, an assign-merge table entry is made or an existing entry is modified. DCB assignments are specified by the two-letter codes in Table 3.

DEVICE OPTIONS

SET commands specifying device options may be issued only between job steps. The device options take effect on subsequent input or output through the DCB. The options are then in effect from job step to job step until reset.

The device options allowed for the SET commands are listed in Table 4. Options corresponding to the M:DEVICE options PAGE, FORM, SIZE, and HEADER are not provided.

FILE OPTIONS

When a DCB is assigned to a RAD storage file or to a labeled tape, certain options may be specified. These options are the same as those that may be specified by a batch ASSIGN command with a few exceptions. Batch ASSIGN options that are not allowed in a SET command include:

1. READ and WRITE - account numbers (default applies).
2. Multiple INSN and/or OUTSN - serial numbers.
3. RECL - record length.
4. TRIES - recovery tries.
5. KEYM - key maximum.
6. VOL - volume number.

Options that are allowed are listed in Table 5.

Table 3. DCB Assignment Codes – SET Command

Type	Codes	Description
Operational Label	BI, BO, C, CI, CO, DO, EI, EO, GO, LL, LO, OC, PO, SI, SL, SO, UC	When the DCB is assigned to one of the system operational labels, the actual device connected to the DCB is that implied by the operational label, if any, for on-line mode.
	NO	No assignment, i.e., no default is to be applied.
Device	CP	Card punch.
	PP	Paper tape punch.
	LP	Line printer.
Magnetic Tape	9T	9-track tape.
	7T	7-track tape.
	MT	Any magnetic tape.
Secondary	DC	RAD data file. (This is the default code if no other code is given.)
	DP	Disk pack storage.

Table 4. Device Options - SET Command

Format	Description
TAB = tab[,tab]...[,tab]	Specifies simulated tab stops and is followed by a list of up to 16 decimal numbers, separated by commas, giving the column position of the stops. If all 16 stops are not specified, the stops given are assigned to the first stops and the remainder are reset.
LINES = value	Gives the number of printable lines per page and is a single decimal value. The maximum value is 255.
SPACE = value	Gives the number of lines of space after printing and is a single decimal value. Values of 0 or 1 result in single spacing. The maximum value is 255.
DRC, NODRC	Turns the special formatting of records on and off. DRC specifies that the Monitor is not to do special formatting of records on read or write operations. NODRC specifies the Monitor is to do special formatting. If neither DRC nor NODRC is specified, NODRC is assumed by default. DRC used in conjunction with BIN will invoke the transparent mode. (See Transparent Mode section of Chapter 10.)
VFC, NOVFC	Controls the formatting of printing by using the first character of each record. VFC specifies that the first character of each record is a format-control character. NOVFC specifies that records do not contain a format-control character. NOVFC is assumed by default.
COUNT = value	Turns on page counting and specifies the column number at which the page number is to be printed.
BCD, BIN	Controls the binary-BCD mode for device read and write operations. BIN used in conjunction with DRC will invoke the transparent mode. (See Transparent Mode section of Chapter 10.)
FBCD, NOFBCD	Controls the automatic conversion between external Hollerith code and internal EBCDIC code (FORTRAN BCD conversion). NOFBCD is assumed by default.
PACK, UNPACK	Controls the packed or unpacked mode of writing 7-track tape. PACK is assumed by default.
DATA = value	Controls the beginning column for printing or punching and is a decimal value. The maximum value is 144.
SEQ = value	Specifies that sequence numbers are to be punched in columns 77-80 of punched output. Four characters of nonblank sequence identification may be given for columns 73-76. Fewer than 4 characters are left-justified and filled with blanks.
L, NOL	Identifies the device type. L specifies that the device must be listing type. NOL specifies that it need not be listing type. NOL is assumed by default.

Table 5. File Options - SET Command

Type	Format	Description
Organization	CONSEC KEYED RANDOM	Consecutive record organization. Keyed record organization. Contiguous relative - sector addressed organization.
Access	SEQUEN DIRECT	Records will be accessed sequentially. Records will be accessed by key.
Function	IN OUT INOUT OUTIN	File is read only. File is write only. File is to be updated. File is scratch.

Table 5. File Options – SET Command (cont.)

Type	Format	Description
Disposition	REL SAVE	File is to be released on closing. File is to be saved on closing.
Size	RSTORE = value	Specifies the number of granules allocated to the RANDOM file.
Storage Control	CYLINDER	Specifies that the data blocks of a public file are to be allocated from public disk packs having cylinder allocation.
Key Storage	NOSEP	Specifies that index blocks of a public file are to be allocated in the same manner as data blocks. (Disk pack if possible; otherwise RAD).
Expiration	EXPIRE = { mm,dd,yy / ddd / NEVER }	Specifies either an explicit expiration date, the number of days to retain the file, or that the files is never to expire.

DETERMINING ON-LINE USER STATUS

The current accounting records applying to an on-line session can be displayed by entering the following command into a terminal:

STATUS

Output is similar to that produced at log-on time and includes:

1. CPU time in minutes and ten-thousandths of a minute.
2. Console time in hours and minutes.
3. Number of interactions.
4. Total charge units.

The format of output is

CPU=M.MMMM CON=h:mm INT=nn CHG=xxxx

LISTING SYSTEM LOAD PARAMETERS

System load parameters supply information about current system operation, such as the number of users currently active and the current values of interactive and compute response times. The format of the command used to display this information is

DISPLAY

Output is

USERS = xxxx
ETMF = xxxx
RESPONSE 90% < xxxx MSECS
RADS = xxxx GRANULES

where

USERS is the number of currently active on-line users.

ETMF is the execution multiplier currently relating program CPU time to job throughput time. ETMF is calculated and updated each minute. It is a moving average covering the preceding minute calculated by summing the time spent computing plus the time spent waiting in high priority ready-to-run queues by all users, and dividing by the sum of time spent computing. Note that since the value is averaged over all users, it is only an approximate measure of how much slower a given process will run due to the time-sharing environment.

RESPONSE gives the number of milliseconds that just exceeds the response time of 90 percent of the responses to terminal requests.

RADS gives the number of unused RAD granules that were available in the user's account at the time he logged on.

SETTING SIMULATED TAB STOPS

Simulated tab stops for a terminal are set by the TABS command. The format of this command is

TABS s[,s]...[,s]

where s is a column position where a tab stop is to be placed.

Up to 16 tabs, in ascending sequence, may be set. Whenever a tab character is sent to or received from a terminal, spaces are sent to the terminal to position the carrier to the next stop that is higher than the current position (if tab simulation is in effect). The setting applies until superseded by another TABS command or by an M:DEVICE procedure call in a program. (The tabs are set in the M:UC DCB.)

CHANGING TERMINAL TYPE

Whenever the type of terminal used with UTS is changed from the type specified at SYSGEN time, UTS must be informed. The system uses this information to adjust character tables and in responses to line-delete and

character-delete options. The format of the command used to identify the terminal type is

TERMINAL type

where type may be any one of the following:

33	for Model 33 Teletype.
35	for Model 35 Teletype.
37	for Model 37 Teletype.
7015	for XDS Model 7015 Keyboard/Printer.

CHANGING TERMINAL PLATEN SIZE

Unless a special platen width and page length are specified, a width of 72 characters and a printable page size of 54 lines are used for all input and output. A width or page length greater or less than this may be specified, if desired. The format of the command that accomplishes this is

PLATEN[w][,l]

where

w is the maximum number of characters to be written per line on the terminal. If more than w characters are written, a line feed and carriage return character sequence is inserted to break up the output into segments no longer than specified by w. If w is 11 or less, no line feed and carriage return sequence is supplied. If the w field is omitted, the current width setting is retained.

l is the number of lines per page of terminal output and must be within the range 0-256. If no l value is given, then the number of lines per page remains unchanged. Note that the automatic page heading and associated spacing results in a standard 11-inch page (54 lines per page) for the default. If l is set to 11 or less, no heading is produced and the page length is unlimited.

Examples:

!PLATEN 72,54 (RET)	sets line width to 72; lines per page to 54.
!PLATEN ,20 (RET)	sets printable lines per page to 20; width remains unchanged.
!PLATEN 27 (RET)	sets width to 27; lines remain unchanged.
!PLATEN ,10 (RET)	turns off page heading; width remains unchanged.
!PLATEN 2 (RET)	prints full line width.

SENDING MESSAGES TO THE OPERATOR

The MESSAGE command causes a message to be sent to the machine operator. The format of the command is

MESSAGE text

the text may be from 1 to 50 characters.

PRINTING OR PUNCHING OUTPUT

Normally the output destined for the line printer and the card punch from all on-line compilations, assemblies, PCL operations, Delta dumps, etc., is accumulated on RAD or push pack until the user logs off. When the user logs off, this output is put in the print and punch queues and is printed or punched when it becomes first in the queue. The PRINT command causes output accumulated for the line printer and punch to be placed in the queue at once. The format of the command is

PRINT

ERROR MESSAGES

During each on-line session, a check is made for a variety of error conditions. Some of these error conditions are detected by TEL, some by BATCH (the BATCH command processor), and some by the Monitor. The messages that are output for these error conditions are listed in Tables 6 and 7, except for Monitor error messages. These are listed in Appendix B.

All error messages are variable and may be changed by the management of an installation through a terminal that is logged on with a special identification and account. The procedure for changing error messages is defined in the UTS/SM Reference Manual, 90 16 74.

TEL ERROR MESSAGES

TEL error messages are all syntax messages. They are listed in Table 6 together with their hexadecimal error subcodes.

BATCH ERROR MESSAGES

Error conditions that may be encountered and reported when a user has submitted a job for batch processing are listed in Table 7. Three categories of error conditions may be encountered: command, job, or system.

TEL COMMAND SUMMARY

Table 8 is a summary of TEL commands. The left-hand column gives the command format, the right-hand column gives the command function and option codes.

BATCH LIMITATIONS

The on-line user's maximum job priority, as specified in his JIT, is the limiting value on the priority he can assign to a batch job submitted on-line, i.e., through the BATCH Subsystem. Also, the maximum values of LIMITS options that he can specify in a batch job submitted on-line are controlled, by the BATCH subsystem, in relation to the actual priority assigned to the job. Table 9 shows these limiting values.

Table 6. TEL Error Messages

Message	Error Subcodes	Description
ASSIGN LIMIT EXCEEDED	5C	The number of DCBs assigned exceeds 12.
BAD PLIST – RESPECIFY DCB	67	The specified DCB was not properly defined by a previous SET command because of a machine software error. For example, the second SET command below would yield an error if the first SET command failed to assign the DCB. !SET M:LO LO ⏎ !SET M:LO; TABS = 5, 10 ⏎
CANNOT ACCESS THE FILE	69	If this message is returned for a PASSWORD command, it indicates that TEL cannot read the user's file because it is open. If the message is returned for a DELETE command, it indicates that no password was specified or that the file to be deleted is in another account.
COMMAND LEGAL AT JOB STEP ONLY	66	The command can be issued in between job steps only.
CONFLICT WITH DELTA – TRY LATER	75	A conflict in use of the M:XX DCB exists.
CONTINUE WHAT?	65	The CONTINUE command can be issued only when a major operation, a subsystem, or executing user program has been stopped or interrupted. An attempt to use it at other times such as between job steps will result in an error message.
DCB NOT ASSIGNED	71	The SET command cannot be used to update a DCB that has not been assigned.
FILE: ME ILLEGAL	60	The terminal may not be used for the requested purpose.
GET WHAT?	73	TEL cannot find the GET file.
IMPROPER FILE BAD DCB BAD JIT BAD LIMITS	 76 77 78	The specified GET file cannot be used because of illegal format.
IMPROPER FORMAT FOR SET CMD	64	A format error has been made in the SET command.
INPUT ERROR - RETRY	5E	TEL received a parity error in the input from the terminal.
INSUFFICIENT ASSIGN/MERGE ENTRY SIZE	70	The total size of all DCB assignments is too large.
NO SUCH FILE IN YOUR ACCOUNT	6A	The file specified by the DELETE command does not exist.
lmn NOT FOUND	62	The load module specified by the START command does not exist.
ON FILE fid ILLEGAL	5F	The file following the preposition ON already exists.
dopt OPTION ILLEGAL FOR DEVICES	6C	The option named in the SET command is not applicable to the device. Only the first non-applicable option is identified (dopt).
fopt OPTION ILLEGAL FOR FILE	6D	The option named in the SET command is not applicable to the file. Only the first non-applicable option is identified (fopt).

Table 6. TEL Error Messages (cont.)

Message	Error Subcodes	Description
PASSWORD CHANGE SUCCESSFUL	6B	The change specified by the PASSWORD command has been made.
PROCESS NOW ACTIVE: QUIT OR CONTINUE	6F	The last command was issued during a Y^C interrupt and would abort the previous command if executed. For example, assume a LINK command is interrupted, !LINK A, B ON E (RET) Y^C ! FORT4 AA ON BB (RET)
QUIT WHAT?	6E	QUIT is legal only in a "break" condition. An error message is returned if the command is issued in between job steps.
SAVE WHAT?	74	TEL cannot find the SAVE file.
START WHAT?	63	Either the START command did not specify a load module or it specified a dollar sign and there was no previous link operation.
TERMINAL TYPE NOT VALID	72	The TERMINAL command specified a terminal type other than 33, 35, 37, or 7015.
UNABLE TO READ A/M TABLE	50	TEL could not get the I/O devices necessary to read the assign-merge table during the job step. The message indicates there is something wrong with RAD storage or the software.
WHAT FID?	68	The name of the file was not specified by the DELETE command.

Table 7. Batch Service Error Messages

Type	Message	Description
Command	ACCESS ERROR	The file is in use or has been assigned a password and no password has been specified.
	EH? @ n	A syntax error exists at character n.
	NO SUCH FILE	The file or account does not exist.
Job	COMMAND REJECTED	The file contains a BIN or FIN control command.
	DATA LOST	The job expects card image input: 80 characters-per-record maximum, EBCDIC; 120 characters-per-record maximum, binary.
	EH? @ n	A syntax error exists at character n.
	ILLEGAL PRIORITY	The terminal-batch job priority may not exceed the user's maximum on-line priority. This maximum value is contained in the user's job-information-table (JIT).
	ILLEGAL NAME	The name on the JOB control command must match the user log-on name.

Table 7. Batch Service Error Messages (cont.)

Type	Message	Description
Job (cont.)	ILLEGAL ACCOUNT	The account on the JOB control command must match the user log-on account.
	ILLEGAL KEYWORD	A limit parameter keyword not corresponding to a recognized system resource was specified.
	ILLEGAL VALUE	A specified limit-option value exceeded the maximum value implied by the specified job priority (see Table 9); or the value was not a decimal integer.
	MISSING JOB COMMAND	The first record of the job must be a JOB control command.
System	BATCH QUEUE FULL	No more symbiont space is available or the queue is full.
	FILE READ ERROR	A file read error occurred; the job must be restarted.

Table 8. TEL Command Summary

Command	Description
BACKUP fid	Saves the specified file on a system tape. In case of a crash in which files are lost, files on the tape will be restored.
BATCH fid	Enters the specified file in the batch job stream.
BUILD fid	Accepts a new file from the terminal.
COMMENT {ON / OVER} list	Directs error commentary to the specified device. Options: list may be fid, LP, or ME.
CONTINUE	Continues processing from the point of interruption.
COPY sf {TO[t] / OVER} df (Simplified format)	Copies a file or device input to the specified file or device. Options: sf may be fid or device code. df may be fid or device code. (See PCL section for complete description.)
DELETE fid	Deletes the specified file.
DELTA	Calls the Delta subsystem.
DISPLAY	Lists the current values of various system parameters.
DONT COMMENT	Stops error commentary output.
DONT LIST	Stops listing output.
DONT OUTPUT	Stops object output.
EDIT fid	Calls Edit to modify a file.

[t]Whenever TO is specified, ON may be substituted.

Table 8. TEL Command Summary (cont.)

Command	Description
END	Terminates the current job step.
FORT4[sp] [ON / OVER [rom][,list]]	Compiles an XDS Extended FORTRAN IV source program. Options: sp may be fid or ME. rom may be fid only. list may be fid, LP, or ME. Output may be interrupted and continued by the following commands: LIST / DONT LIST OUTPUT / DONT OUTPUT COMMENT / DONT COMMENT CONTINUE
GET fid	Restores the previously saved core image.
GO	Continues processing from the point of interruption.
JOB jid	Requests the status of remotely entered jobs.
LINK [codes]rom[,rom]...[,rom] [ON / OVER lmn] [;lid[,lid]...[lid]] [UNDER FDP]	Forms the load modules as specified. Options: library search: (L), (NL), (Pi), (FDP), (NP) default: (L), (P1) display: (D), (ND), (C), (NC), (M), (NM) default: (D), (C), (NM) symbol tables: (I), (NI) default: (I) rom may be fid or $; parentheses enclosing roms cause merge of symbol tables. lid must name a file containing one or more ROMs.
LIST [ON / OVER list]	Directs the listing output to the specified device, or counteracts the preceding DONT LIST command. Options: list may be fid, LP, or ME.
lmn [sp] [ON / OVER [rom][,list]]	Initiates execution of a load module. Options: lmn has the form: name[.[account][.password]] absence of period and account specifies system account. presence of period and absence of account specifies log-on account. sp is assigned to M:SI DCB. rom is assigned to M:GO DCB. list is assigned to M:LO DCB.

Table 8. TEL Command Summary (cont.)

Command	Description
MESSAGE text	Sends the specified message to the operator.
META[sp] [ON / OVER [rom][,list]]	Assembles the specified source program. Options: sp may be fid or ME. rom may be fid only. list may be fid, LP, or ME. Output may be interrupted and continued by the following commands: LIST, OUTPUT, COMMENT, DONT LIST, DONT OUTPUT, DONT COMMENT, CONTINUE
OFF	Disconnects terminal from system and provides accounting summary.
OUTPUT [ON / OVER rom]	Directs object output to the specified device, or counteracts the previous DONT OUTPUT command. Options: rom may be fid only.
PASSWORD xxxx	Assigns a new log-on password for the user. xxxx is 1-8 characters. Any of the following characters may be used: A-Z a-z 0-9 _ $ * % : # - @ backspace.
PLATEN [w][,l]	Sets the value of the terminal platen width and page length.
PRINT	Sends print output to the line printer and punch output to the punch.
QUIT	Terminates the current job step.
RUN [codes][rom [,rom]...[,rom] [ON / OVER lmn] [;lid[,lid ...][,lid]] [UNDER DELTA / FDP]	Loads the specified module and starts execution. Options: library search: (L), (NL), (Pi), (FDP), (NP) default: (L), (P1) display: (D), (ND), (C), (NC), (M), (NM) default: (D), (C), (NM) symbol table: (I), (NI) default: (I) rom may be fid or $; parentheses enclosing roms cause merge of symbol tables. lid must name a file containing one or more ROMs.
SAVE {ON / OVER} fid	Saves the current core image on the designated file.

Table 8. TEL Command Summary (cont.)

Command	Description
SET dcb 0 SET dcb [oplabel / device / tapecode [tapeid]] [;dopt[;dopt]...[;dopt]] SET dcb [tapecode[tapeid]/fid / filecode[packid]/fid] [;fopt[;fopt]...[;fopt]]	Assigns file or device to a DCB or sets DCB parameter. Options: see Tables 3, 4, 5, and 6.
START [lmn / $] [UNDER DELTA]	Begins with execution of the program just loaded, either with or without an associated debugger.
STATUS	Displays the current accounting values.
Subsystem Calls BASIC FORT4 CONTROL META DELTA PCL EDIT SUPER lmn (user's program)	These calls are entered while TEL is in control of the terminal. They turn over control of the terminal to the subsystem.
TABS s[,s]...[,s]	Sets the simulated tab stops at the terminal.
TERMINAL type	Sets the terminal type for proper I/O translations. Type may be 33, 35, 37, or 7015.

Table 9. BATCH Subsystem Limits - Option Maximums Versus Job Priority

Priority	TIME min	PO cards	LO pages	DO pages	UO pages	TSTORE granules	PSTORE granules	SCRATCH tapes
0	9999	9999	9999	9999	9999	9999	9999	50
1	9999	9999	9999	9999	9999	9999	9999	50
2	240	9999	9999	9999	9999	9999	9999	50
3	120	9999	9999	9999	9999	9999	9999	50
4	60	9999	9999	9999	9999	9999	9999	50
5	30	9999	9999	9999	9999	9999	9999	50
6	15	9999	9999	9999	9999	9999	9999	50
7	5	9999	9999	9999	9999	9999	9999	50
8	5	9999	9999	9999	9999	9999	9999	50
9	5	9999	9999	9999	9999	9999	9999	50
A	5	9999	9999	9999	9999	9999	9999	50
B	5	9999	9999	9999	9999	9999	9999	50
C	5	9999	9999	9999	9999	9999	9999	50
D	5	9999	9999	9999	9999	9999	9999	50
E	5	9999	9999	9999	9999	9999	9999	50
F	5	9999	9999	9999	9999	9999	9999	50

†The maximum batch-job priority for on-line initiation is seven in the system as-distributed (set in the user's JIT). Therefore, the portion of the table below the dashed line is not utilized. Both the maximum priority and the table values can be altered by the user-installation.

4. META-SYMBOL, EXTENDED FORTRAN IV, AND BASIC OPERATIONS

INTRODUCTION

Meta-Symbol, Extended FORTRAN IV, and BASIC processors may be used under UTS in either on-line or batch mode. The on-line operating features of these processors are described in this chapter. The batch operating features are described in the following manuals:

Meta-Symbol/LN, OPS Reference Manual, 90 09 52

Extended FORTRAN IV/OPS Reference Manual, 90 11 43

BASIC/Reference Manual, 90 15 46

META-SYMBOL

The Meta-Symbol assembler is called from an on-line terminal by the following command:

META [sp] [ON / OVER [rom] [,list]]

where

sp specifies a source program and may be either a file identification (fid) or the terminal identification (ME). If no source file is specified, TEL assumes input is from the file/device currently assigned to the M:SI DCB. If the M:SI DCB is not assigned, TEL expects input to come from the terminal (ME). (Note that on-line DCB assignments are made explicitly by the SET command and implicitly by META. Once set, DCB assignments remain in effect until reassignment by subsequent SET commands or specified META options.)

ON indicates that ROM output is to be on a new file.

OVER indicates that ROM output is to be over an existing file or on a new file.

rom specifies that the relocatable object module produced by assembly is to be directed to a specific file (fid). If no ROM is specified, output is directed to a special file that may subsequently be referenced by a dollar sign. (rom is assigned to the M:GO DCB.)

list specifies that listing output is to go to a file (fid), a line printer (LP), or the terminal (ME). If list is not specified, TEL assumes that the listing output is to go to the file/device currently assigned to the M:LO DCB. If the M:LO DCB is not assigned, TEL produces no listing output.

This command replaces the control cards that are needed to perform the equivalent operations through batch processing. The replaced cards are

```
!JOB...
!ASSIGN M:SI...
!ASSIGN M:LO...
!ASSIGN M:GO...
!METASYM SI,LO,GO
```

When the assembler is entered, it sends a request for options (WITH>) to the terminal. If there are no options, a carriage return character may be entered following the request. This initiates the assembly, providing additional inputs are not required by the assembler.

Example:

Assume that file A is to be assembled with ROM output going to B and list output going to the terminal. No special assembly options are desired, and no additional input is required by the assembler.

```
!META A ON B,ME (RET)
WITH> (RET)
*       ERROR SEVERITY: 0
* NO ERROR LINES
!
```

If assembly options are desired, the codes (Table 10) for the desired options are entered following the request for options. These codes are separated by commas and terminated by a carriage return or line feed character which initiates assembly. If a concordance option (CN in Table 10) has been specified, additional input is required. Meta-Symbol sends a prompt character to the terminal to request each concordance command. Assembly is initiated only after the last concordance control command (.END) has been entered.

Some of the assembly options available in batch mode are not recommended in on-line mode. These options either have no meaning for on-line mode or are assumed when the META command is used (such as GO, LO, and SI). Options that are allowed are listed in Table 10.

Examples:

1. Assume that a RAD storage file, called SOURCE, is to be assembled. ROM output is to go to BIN and list output is to go to the line printer. A cross-reference is to be included with list output. The cross-reference is to exclude symbols X1 and X2 and to include operation code CAL3.

```
! META SOURCE ON BIN, LP (RET)
WITH> CN (RET)
> .SS X1,X2 (RET)
> .IO CAL3 (RET)
> .END (RET)
```

Table 10. Meta-Symbol Assembly Options[t]

Option	Description
AC (ac_1,ac_2,...,ac_n)	Specifies alternate accounts that are to be searched when the assembler must access system files that are not logged either under the system (:SYS) account or under the user's log-on job account. The ac items are alternate account that are searched first; then by default, the :SYS account and finally the log-on account are searched as necessary.
DC	Specifies that a "standard" concordance is to be produced on the LO device. The "standard" listing does not include operation code names, but otherwise includes all symbol references, including function and command procedure names and intrinsic functions.
CN	Requests that a symbolic cross-reference listing be included with the assembly listing. When this option is given, the assembler sends a prompt character to the terminal to indicate that concordance control records identifying special concordance options should be entered. One control record, preceded by a period, is entered following each prompt character. The last control record must be an END record. The concordance control commands are as follows: IO — Include all or a selected set of operation codes. SS — Suppress all or a selected set of symbols. OS — Include only a selected set of symbols. DS — Produce a modified LS listing, displaying only lines that reference a selected set of names. END — Terminate concordance control commands.
CO	Causes the assembler to produce a compressed version of the input program on the file specified in the M:CO DCB. This DCB must have previously been assigned by a SET command.
LU	Requests that the assembler include a listing of the Meta-Symbol update records with the program listing.
NS	Requests that no assembly summaries be included with the listing.
SD	Causes the assembler to produce symbolic debugging object code for use with the Delta debugging processor. The object code is included with the standard binary output ROM.
SO	Causes the assembler to create a source output file corresponding to the input program. The input program may be Edit-source, compressed, or compressed with updates. The M:SO DCB must have been previously assigned.
CI	Causes the assembler to access M:CI for compressed input. Typically it would be specified if the user wishes to update the compressed file with the contents of the source file assigned to M:SI (via, e.g., the META command). The source input on the M:SI file must be terminated with a +END statement. The M:CI DCB must have been previously assigned by a SET command. Consult the Meta-Symbol/LN, OPS Reference Manual, 90 09 52 for a full discussion of the assembler's operation when both SI and CI inputs are specified.

[t]For additional details concerning assembly options, refer to the Meta-Symbol/LN, OPS Reference Manual, 90 09 52.

2. Assume that a source program, called SOURCE, is to be assembled with ROM output going to BIN and listing output going to the line printer. The following assembly options are desired:

   a. A source output file (SOURCEOUT) corresponding to SOURCE.

   b. A compressed version of SOURCE.

   c. A symbolic cross-reference.

   d. A symbolic debugging object code for Delta.

```
!SET M:SO DC/SOURCEOUT (RET)
!SET M:CO CP (RET)
!META SOURCE ON BIN,LP (RET)
WITH> SO,CO,CN,SD (RET)
>.END (RET)
```

When input is from a keyed Edit file, a decimal representation of the sequence number for each record is placed in the assembly listing. This representation is placed in the position normally occupied by columns 73-80 of an input card.

It is possible to make use of Meta-Symbol's internal editor in conjunction with compressed source files while running on-line. The internal editor and source compression facility are oriented toward card image batch processing but can be useful to on-line operation when backup files must be kept on cards or when work must be done in strictly a BPM-compatible fashion. These Meta-Symbol features are described in Chapter 12 of the Meta-Symbol/LN, OPS Reference Manual, 90 09 52.

If a program in compressed format exists on RAD or disk pack storage, either as the output of the assembler or as a result of a file management operation, it can be assembled with on-line Meta-Symbol simply by specifying it as input (sp) in a META command. Meta-Symbol distinguishes between the keyed source format of the Edit files and the sequential binary format of compressed files.

Example:

Assume that CI-FILE is a program file in compressed format. This file is to be assembled with ROM output going to BO-FILE and list output going to the line printer.

```
!META CI-FILE ON BO-FILE, LP (RET)
WITH> (RET)
```

It is also possible to maintain an update file through the use of the Edit subsystem and to use the file to modify a compressed file. In this case the former would be assigned to M:SI and the latter to M:CI, via a SET command and the CI assembly option.

Example:

Assume that an update file (UPDATE-FIL) is being maintained under Edit and is to be used to update a CI-FILE on labeled tape. ROM output is to go to BIN and list output is to go to the line printer.

```
!BUILD UPDATE-FIL (RET)
  1.000 + 4,6 (RET)
  2.000          BANZ EXIT (RET)
  3.000 + 10,10 (RET)
  4.000 + END (RET)
  5.000 (RET)
*END (RET)
!SET M:CI LT#1234I/CI-FILE (RET)
!META UPDATE-FILE ON BIN,LP (RET)
WITH>CI (RET)
```

FORTRAN IV

The XDS Extended FORTRAN IV compiler is called from an on-line terminal by the following TEL command:

FORT4 [sp] [ON / OVER] [rom] [,list]

where

sp specifies a source program and may be either a file identification (fid) or the terminal identification (ME). If no source file is specified, TEL assumes input is from the terminal (ME). (sp is assigned to the M:SI DCB.)

ON indicates that ROM output is to be on a new file.

OVER indicates that ROM output is to be over an existing file or on a new file.

rom specifies that the relocatable object module produced by compilation is to be directed to a specific file (fid). If no ROM is specified, output is directed to a special file that may subsequently be referenced by a dollar sign. (rom is assigned to the M:GO DCB.)

list specifies that listing output is to go to a file (fid), a line printer (LP), or the terminal (ME). If list is not specified, ME is assumed, but no listing output is produced until a LIST command is issued (list is assigned to the M:LO DCB).

The naming of files sp, rom, and list can be thought of as simple assignments for the DCBs used by the compiler. The DCBs M:SI, M:GO, and M:LO are used by FORTRAN for its input and output operations and UTS directs the data to and from the respective files. The specifications sp, rom, and list are used for these assignment purposes and have no effect on the operation of the compiler. The control of the compilation rests with the compiler options described below.

In the absence of a specification for rom or list, UTS will direct the data to or from the last file or device to which GO or LO was assigned. In this way a user may make these assignments at the beginning of a job and they will remain in effect until changed. If an identifier is not specified for ROM, the object program produced by the compilation will be written on a scratch file which may be referenced later by the name $.

When the FORTRAN IV compiler is entered in the on-line mode, it sends a request for options to the terminal by typing

OPTIONS>

The user may then enter the option codes (Table 11) to be used for this compilation. The codes are separated by commas and terminated by a carriage return. If no option codes are entered before the terminating carriage return, the compilation will be done as though the single option PS had been typed. The PS option ensures that the user is aware of the size of his program and the first and last card while producing a minimum of output. A source input file is always expected and an object program is always produced when operating from an on-line terminal.

Table 11. FORTRAN IV Compilation Options[t]

Option	Description
ADP	Causes all real operations to be done in double precision and all complex operations to be done in double complex. (See Extended FORTRAN IV/LN Reference Manual, 90 09 56.)
BC [(n)]	Permits a number of programs to be compiled from the source file. When this option is used, the compiler reads source programs until the conditions of the option are met. Thus, a number of different programs may be compiled using only one FORT4 command. The suboption, n, allows the BC option to specify compilation of the first n programs from the source file.
BO	Causes a binary object deck to be produced (via M:BO). If the BO option is used, the correct assignment for M:BO must be ensured. There is no default assignment for this DCB.
DEBUG	Causes the compiler to generate linkages, such as internal symbol tables, to the FORTRAN Debug Package.
GO	This option is redundant. In on-line operation, a binary object deck is produced for all programs via the M:GO DCB.
LO	Lists the object program on the LO device.
LS	Lists each source program and compilation summary on the LO device.
NMP	Causes the generated code of the object program to be a control section with protection type 00 instead of 01.
NS	Eliminates the compilation summary map and the printing of the first and last card of the source program. To eliminate the entire listing of a compilation, NS or PS should be specified and LS or LO should not be specified.

[t]For more details concerning compilation options, refer to the Extended FORTRAN IV/OPS Reference Manual, 90 11 43.

Table 11. FORTRAN IV Compilation Options[†] (cont.)

Option	Description
PS	Causes the first and last cards and a partial summary map of the program to be printed. The partial summary map includes NUMBER OF ERROR MESSAGES:n NUMBER OF STATEMENTS DELETED:m } These are printed only if there were errors in the program. HIGHEST ERROR SEVERITY: { 0 (NO ERRORS) / 4 (NO MAJOR ERRORS) / 7 (MAJOR ERRORS) / 10 (MAJOR ERRORS) } DEC WORDS / HEX WORDS GENERATED CODE: d d d d d / x x x x x CONSTANTS: d d d d d / x x x x x LOCAL VARIABLES: d d d d d / x x x x x TEMPS: d d d d d / x x x x x TOTAL PROGRAM: d d d d d / x x x x x
S	Specifies that in-line assembly code is to be accepted on cards that have an S in column 1. For information concerning the rules for in-line symbolic code, see the Extended FORTRAN IV/LN Reference Manual, 90 09 56.
SBIT	Preserves the integrity of the maximum negative number expressible on a Sigma computer.
SI	Specifies source input. This is unnecessary but is acceptable for compatibility.
SO	Reproduces the source program on the source output file (via M:SO).
X	Compiles records with X in column 1.

[†]For more details concerning compilation options, refer to the Extended FORTRAN IV/OPS Reference Manual, 90 11 43.

After the option request has been completed, the compiler reads the source program from the sp file. Input continues until an END statement or end-of-file (ESC F keys) is encountered. The program summary and object program are then output as requested and control is returned to the UTS executive (TEL). If the source file contains more than one program, subsequent compilations can be obtained from it by using the BC option.

When used from an on-line terminal, the compiler accepts horizontal tab characters in source program records. It replaces each tab character with the correct number of spaces to locate the next input character at the position specified by the next tab stop. At the on-line terminal, this positioning is done by the UTS executive so that the typist is aware of the tabbing action. Internally, the compiler performs a similar action by inserting the correct number of spaces into the source record image. Any characters in the record following the tab character are shifted to the right. The listing output and source output from the compiler do not contain the character. They contain the spaces which were inserted into the image. As many tab characters as are required may be entered but care should be taken to ensure that tab stops are provided. If the compiler cannot match a tab character

with a corresponding tab stop position during its internal expansion operation, the tab character will remain in the record and will cause a syntax error. Tab stops may be set from the on-line terminal by the TEL commands TAB and SET.

When accepting source input from an on-line terminal, the compiler normally checks each record as it is typed and immeditaely prints out any diagnostic on the following line. However, if a statement is to be continued, this error checking is not done until the continuing statements have all been input. When a statement is to be continued, the last character preceding the carriage return must be a colon (:) to indicate that this record is continued. The next record, the continuation record, must follow standard FORTRAN rules and have blanks in columns 1 to 5 and a continuation character in column 6. Statements containing errors and continued over several records have their error diagnostics printed following the last record. The colon used to indicate that a record is continued is removed from the record and replaced with a blank character.

Since the colon is contained in the Extended FORTRAN IV standard character set, it is possible to use it in a FORTRAN statement. Some difficulty might be expected in statements which end with a colon, such as, A = 4HABC:. This problem can be overcome by typing an extra blank following the colon and before the carriage return.

Examples:

Assume a program is to be compiled with the source input read from file SOURCE, the relocatable object module written onto file DECK, and the listing written onto the user's terminal.

```
! FORT4 SOURCE ON DECK,ME (RET)
OPTIONS>LS(RET)
```

Since the compiler always expects an SI file and always generates a GO file, the only option (LS) is used to cause a listing of the source program at the user's terminal. If any errors occur they are printed at the terminal.

UTS BASIC

UTS BASIC may be operated in on-line or batch mode. The on-line mode is expected to be normal. Batch operations are limited to those requiring no user intervention and differ from on-line operations primarily in the assignment of input/output devices.

The BASIC system is called from a UTS terminal in the following way:

```
! BASIC (RET)

>
```

When the system is ready to accept input, it prompts with a "greater than" character (>). At this point, BASIC is in editorial mode with no program text.

In the on-line mode, BASIC returns to TEL only if terminal input failure occurs, the BREAK key is activated twice without any intervening terminal input or the SYS [TEM] command is typed. In batch mode, exit to the Monitor also occurs after a compilation that contains errors, or after a run-time error.

While using BASIC in on-line mode, the user fully controls the flow of activity via the terminal. The normal mode for doing so is to respond to prompt characters that indicate the system is prepared for input. Two prompt characters are used: a question mark and a "greater than" symbol. A question mark indicates that execution is in progress and input data is required. A "greater than" symbol indicates that the system is ready for editorial input or commands.

In some instances, such as during the output of an extended listing or when a program is suspected of being in a loop, it is desirable to acquire terminal control without waiting. A facility is provided via the BREAK key activation to interrupt current activity.

For additional detail about BASIC operations, refer to the BASIC/Reference Manual, 90 15 46 – Revision B or later.

5. PERIPHERAL CONVERSION LANGUAGE

INTRODUCTION

The Peripheral Conversion Language (PCL) is a utility subsystem designed for operation in a batch or on-line environment under UTS. It provides for information movement among card devices, line printers, Teletype terminals, magnetic tape devices, and RAD or disk pack storage.

PCL is controlled by single-line commands supplied through on-line terminal input, through a file containing PCL commands, or through command card input in the batch job stream. The command language provides for single or multiple file transfers with options for selection, sequencing, formatting, and conversion of data records. Additional file maintenance and utility commands are provided. The actual input/output operations are carried out using standard UTS CALs.

For batch operation, PCL is activated by a !PCL control command card in the job stream. Once active, PCL reads subsequent command cards directly through the M:SI DCB until terminated by an END command card or some other control command card. Input and output is done through the M:EI and M:EO DCBs respectively. Error messages are transmitted to the device currently assigned to the M:DO DCB.

For on-line operation, PCL is called by typing "PCL" while TEL is in command of the terminal. PCL responds by typing "PCL version HERE" followed by a prompt character (<) at the left margin of the next line. This indicates that PCL is ready to accept a command.

Example:

```
!PCL (RET)
PCL B00 HERE
<
```

When accepting or processing a command on-line, PCL is in the command state. Entry to this state is always indicated by the display of the PCL prompt character. Once a valid command begins execution, PCL enters the active state. In this state, PCL prompts for input, if required, with a period (.). This state remains in effect until execution of the command terminates, at which time PCL reenters the command state, issues a < prompt character, and waits for the next command. As in batch operation, user input and output is processed through the M:EI and M:EO DCBs; error messages go to the M:UC DCB and commands are received through the M:SI DCB.

The user has the option of building a file of PCL commands and having the commands executed by preceding the call to PCL by an ASSIGN or SET command that assigns M:SI to the file of commands. In this case, PCL will not prompt the on-line user for input, but will print each command, preceded by a prompt character (<), as it begins execution of the command.

Example:

```
!SET M:SI/CMDFILE (RET)
!PCL (RET)
PCL B00 HERE
< first command
.
.
.
```

The following description of PCL is oriented toward the on-line user. For the batch user, communication is established with input through the job stream and output through the M:LO DCB with no user interaction. Thus, all user prompting and terminal-specific operations given here may be ignored by the batch user.

CONVENTIONS

SYNTAX

PCL is a free form language with a few restrictions imposed for simplicity in implementation and use. These restrictions are outlined below:

1. All commands must comply with the general format given in the definition.
2. Blanks preceding or following an argument field are permitted; embedded blanks are not permitted.
3. At least one blank must follow each command verb, except REW and REM when followed by a number (#) character, and must precede and follow each command preposition (TO, ON, or OVER).
4. Continuation between input records is not allowed. (Only one command per line is allowed.)
5. "End-of-command" is indicated by the end of the input record (column 72) for card input or by a carriage return or line feed character for either card or Teletype input.
6. Only one input device and only one output device may be open at any given time.

DEVICE IDENTIFICATION CODES

Device identification codes are symbols used to identify source and destination devices in PCL commands. These codes are listed in Table 12.

Table 12. Device Identification Codes

Device Code	Description
CR	Card reader (Not available for on-line operations. For batch operations, files are separated by two successive EOD control cards.)
CP	Card punch.
LP	Line printer.
ME	On-line terminal (Input is terminated by an ESC F – end-of-file – code.)
DC	RAD storage.
DP	Disk pack.
LT	Labeled tape.
FT	Free form tape. (Files are separated by an EOF mark.)

Most device codes correspond to unformatted unit record equipment, and the action is very close to direct device access. In the case of the codes DC, DP, and LT, however, the intent is not to give access to RAD, disk pack, and magnetic tape as devices, but to provide a means for symbolic reference to files of information created and maintained by the Monitor's file management system.

Through various subsystems and/or Monitor services, the user can create logically connected groups of records called files. Each file has a name by which it is known. These files are contained on RAD (DC), disk pack (DP), or labeled magnetic tape (LT). Labeled tape carries internally a serial number and the creator's account number, in addition to the file names and the file contents.

The following paragraphs give the conventions to be used when creating or otherwise working with files.

FILE AND REEL IDENTIFICATION

A file identifier (fid) has three parts: name, account, and password. A file name consists for PCL of 1 to 31 characters,[t] which in general may be any characters except the following PCL delimiters:

blank . ( ) ; / ,

However, any character including these delimiters may be used in a file name if the name is delimited by single quotes, e.g., '(A)'. Single quotes within such a file name must each be represented by paired quotes.

A hexadecimal format may be used to represent a file name that contains one or more unprintable characters, e.g., X'00E7'.

When PCL outputs a file name, account, or password, it prints the string in hexadecimal format if any of the characters do not belong to the EBCDIC 57-character set.

Account and password are one to eight characters from the same set and may also be written as hexadecimal or character strings. The various combinations are written as follows:

name	file in log-on account directory.
name.account	file in specified account directory.
name.password	file in log-on account with password.
name.account.password	file in specified account, with password.

In general, a job may create, delete, read, or modify files in the account in which it is running. However, files in different accounts can only be read – not created, deleted, or modified. A file identifier is the same whether the file is on RAD, disk pack, or labeled tape. However, in order to access a file on labeled tape, the physical reel identifier must in general also be given.

To access a file on a private disk pack, the serial number of the primary volume must be given. When creating files on a disk pack, all serial numbers for the volume set must be specified. The following description of a reel identifier applies to disk pack as well as to labeled tape.

A reel identifier (reel-id) consists of two parts: a serial number and an account number.

The account has the same format as described above, while a serial number is one to four alphanumeric characters of the same character set as file identifier, except that the number sign (#) may not be used. Also, character string and hexadecimal string notation are not allowed. The two permissible forms for a reel identifier are as follows:

#serial no. [#serial no.][#serial no.]

Reel(s) created, or to be created, in log-on account.

#serial no. [#serial no.][#serial no.] . account

Reel(s) created in specific account.

The # is a syntactic identifier used to introduce the serial number, e.g.,

#MEFA
#MEF1#MEF2.C7308300

The optional second and third serial numbers are used to indicate a multi-volume file or set of files.

[t] Note that most on-line subsystems allow a maximum of 10 characters for a file name.

In general, a job cannot create files on a labeled tape or disk pack in a different account than that in which it is executing. However, it may read tapes or disk packs that were created in different accounts.

Therefore, in subsequent command descriptions, the following convention is adopted. If a reel identifier is used in an input sense, where either of the above representations is valid, then it will be symbolized as "#reel-id". However, if it is used in an output sense, where only a serial number is valid, then "#serial no." will be used explicitly. In either case, up to three serial numbers may be specified if a multi-volume file is involved. Free form tape (FT) only needs to be identified by a serial number.

The absence of a reel identifier on a labeled tape or free form tape specification implies that a scratch tape is to be used. After the first occurrence of a scratch tape specification in an output sense, the output serial number of the tape is communicated to the on-line user in order that this tape may be referenced by subsequent commands. However, a reel identifier is not actually required by any command. If a scratch tape is used for the first time in an input sense, an I/O error is reported. If a scratch tape has been written, a command in the same PCL session that specifies a tape without a reel identifier, in either an input or output sense, is interpreted by PCL as referring to the same scratch tape. PCL must be reentered if a second scratch tape is needed.

If the file is random, the absence of a reel identifier on a disk pack specification indicates that the system disk pack is to be used. For other types of files, the absence of a reel identifier causes the DP device code to be treated the same as DC.

CAPABILITIES

The following is a list of available functions in PCL defined in terms of the actual command verbs:

COPY device(s) and/or file(s) TO[†] device or new file.

COPY device(s) and/or file(s) OVER device or existing file.

COPYALL files in specified account on RAD or disk pack TO labeled tape(s) or to a device.

COPYALL files in specified account on RAD or disk pack TO log-on account on RAD.

COPYALL files on labeled tape(s) TO RAD or disk pack.

COPYALL files on labeled tape(s) TO files on labeled tape(s) or to a device.

COPYSTD performs a copy of a control file and all files indicated within the control file.

DELETE an existing file.

DELETEALL deletes all or a portion of the user's RAD files.

LIST a file directory for RAD, tape, or disk pack.

REVIEW user's RAD file directory.

SPF space file ±n files on free form (unformatted) magnetic tape.

WEOF write end-of-file on current output device

REW rewind designated tape.

SPE space to end of last file on labeled tape.

REM remove designated tape or disk pack.

TABS define tab settings for tab expansion.

[†]Wherever TO is specified, ON may be substituted.

BREAK FUNCTION

The function of the BREAK key under PCL (as under TEL) is to interrupt current activities. If the BREAK key is pressed while PCL is in the active state, PCL usually terminates what it is doing, such as printing or copying, passes control to the terminal, and reverts to the command state. If the BREAK key is pressed while PCL is in the command state, PCL ignores the current command as if X^c has been pressed. The effect of the interruption or the termination varies with the command being executed and is discussed in detail with each command, where necessary. If no mention is made of the effect, the BREAK key is assumed to have no effect on execution of the command.

FILE COPY COMMAND

The file COPY command permits single or multiple file transfers to take place between peripheral devices or between file storage and peripheral devices. Options are included for selecting, formatting, and converting data records.

COPY COMMAND FORMAT (GENERALIZED)

The COPY command is of the form

COPY source[source...] [TO / OVER destination]

where

source may be an input device such as card reader (CR), a RAD file (e.g., ALPHA), a file on private disk pack, or a file on labeled or free form tape. File concatenation may be performed by specifying more than one source device or file.

destination may be an output device such as card punch (CP), a RAD file, a file on private disk pack, or a file on labeled or free form tape. Absence of a destination specification is allowed and will normally cause file extension to occur.

If the destination of the COPY is a RAD file currently existing in the user's account directory, PCL will require that the preposition OVER be used in the command. That is, COPY TO or COPY OVER will create a file, but for the user's protection only COPY OVER can replace an existing file. After this check, PCL opens the source devices and files one at a time in the order given, and copies them to the destination device or file. Source files are closed after they have been copied. The destination device or file is closed at the same time.

If the BREAK key is pressed during execution of the COPY command, PCL responds by typing the message 'ENTER X TO ABORT COMMAND'. Any character typed, except X, causes continuation of the command. Typing an X aborts the command.

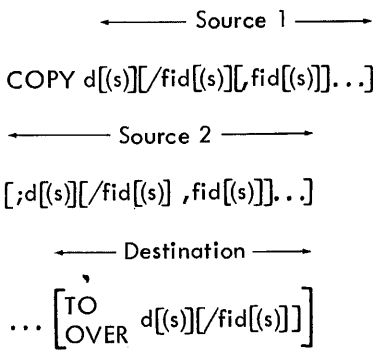
Note that the TO or OVER command preposition and the destination are optional. If the COPY command contains only a source specification, PCL uses the destination device or file defined on the most recently issued COPY command containing a destination specification. (This is illustrated in the sixth COPY example.) It should be noted that file extension will occur in this case. Any PCL command except COPYALL may be used between the COPY defining the destination specification and the COPY with this specification omitted, since the output specification will not be changed by these commands.

If a COPY command is used without a destination specification and a destination has not been defined by a previous command, the default destination is to the terminal.

The message '..COPYING' prints at the terminal when the copy operation begins if neither the input nor the output device is ME.

COPY COMMAND FORMAT (SPECIFIC)

The specific format of the COPY command is



where

d represents device identification and has the form

device identification code[#reel-id]

Device identifications were defined in Table 12. Reel identifiers apply only to magnetic tapes or disk packs (LT, FT, or DP). Absence of a reel identifier for a tape device implies a scratch tape.

/ separates a device identification code from the files on that device.

fid represents file identification and has the form

name[[.[account].password] / .account]

The DC device identification code is optional on a COPY command referencing a RAD file. For example, RAD file A may be specified in one of three formats: DC/A, /A, or A. However, this flexibility makes the device codes in Table 12 reserved words. For example, file CR must be referred to as DC/CR, /CR or 'CR', never simply as CR. The format /A cannot be preceded by device options.

, separates files on the same device.

; separate devices.

(s) represents specifications for data encoding: data codes (Table 13), formats (Table 14), modes (Table 15), and record selection. It has the form

(option[,option]...[,option])

Specifications given at the device level apply to all files on that device. Those given at the file level apply to that file only and have precedence if a conflict occurs between levels.

Data encoding is discussed in detail below.

Examples:

1. Assume that three consecutive files, each terminated by a double !EOD mark, are to be copied from a card reader to an existing RAD storage file called ALPHA. (This would only be allowed in batch.) The PCL command would be:

```
COPY CR;CR;CR OVER DC/ALPHA
```

or

```
COPY CR OVER ALPHA

COPY CR

COPY CR
```

2. Assume that a Meta-Symbol source program file, called SOURCE, is to be copied from RAD storage to the terminal. The command could be coded as

```
<COPY DC/SOURCE TO ME ⏎
START LW, R1 ALPHA
      AI, R1 5
      CW, R1 BETA
```

This command could also be typed as

```
≤COPY SOURCE TO ME ®
```

3. Assume that successive cards are to be copied from the card reader to a new RAD storage file with the following file identification: KD.2024.PLEASE. (This would only be allowed in batch processing.) Two !EODs are used to signal the end of the card file. The COPY command would be:

```
COPY CR TO DC/KD.2024.PLEASE
```

4. Assume that files B and C from labeled tape No. 57 are to be copied, in that order, to a new RAD storage file called B..PASS.

```
≤COPY LT#57/B,C TO DC/B..PASS ®
..COPYING
```

5. Assume file A from labeled tape No. 5, file D from RAD storage, and all files on free form tape No. 8 up to the next double end-of-file are to be copied to file A on labeled tape Nos. 6 and 7. Tape No. 7 is to be used only if No. 6 overflows.

```
≤COPY LT#5/A;DC/D;FT#8 TO LT#6#7/A ®
..COPYING
```

6. Assume three successive sets of files, each separated by a double end-of-file, are to be punched in cards from free form tape No. 7236. Two !EODs are written when the output device is closed.

```
≤COPY FT#7236 TO CP ®
..COPYING
≤COPY FT#7236 ®
..COPYING
≤COPY FT #7236 ®
..COPYING
```

or

```
≤COPY FT#7236;FT#7236;FT#7236 TO CP ®
..COPYING
```

DATA ENCODING

The COPY command may contain various codes and specifications which either describe certain characteristics of input and output files or devices, or which request various types of data conversion or format changes in the output to be produced. Partial files may be copied by use of record selection and output records may have sequence identification inserted or deleted.

A description of the available codes and specifications follows:

DATA CODES

Data codes (Table 13) describe the source or destination data types to be expected or produced.

Table 13. Data Codes

Code	Meaning
E	EBCDIC (default data code)
H	Hollerith

DATA FORMATS

Data formats (Table 14) describe the source or destination record formatting to be expected or produced.

Table 14. Data Formats

Code	Meaning
X	Hexadecimal dump
C	Meta-Symbol compressed

The X option produces a single-spaced dump on the line printer or terminal. The presence of an asterisk following the word count in the dump indicates that omitted lines are identical to the preceding line. If output is to the line printer, the EBCDIC equivalent is also printed.

A C option on an input specification indicates that input is in compressed format and is to be decompressed on output. A C option on an output specification indicates that input is in symbolic form and is to be compressed on output. The presence of a C option on both input and output is invalid. Also, record selection is not allowed when compressing or decompressing files.

MODES

Mode codes dictate the control modes for the specified files or devices. They are shown in Table 15.

Table 15. Mode Codes – COPY Command

Mode	Description
BCD,BIN	Binary-coded decimal or binary mode. These codes are valid for cards, paper tape, and magnetic tape.
7T,9T	7-track or 9-track magnetic tape.
PK,UPK	7-track binary tape packed or unpacked.

Table 15. Mode Codes – COPY Command (cont.)

Mode	Description
SSP,DSP, VFC	Single, double, or variable format controlled spacing on line printer or terminal.
NC	No carriage return. Removes carriage-control character (X'15' or X'0D'), if present, from each record on output. This mode is the default mode if input is from the terminal.
CR	Retains carriage return. Must be specified if carriage returns are to be retained when copying 'ME' to a file or device.
TX	Tab expansion. Values specified on a PCL TABS command are used. If a PCL TABS command was not issued, the tab values in the M:UC DCB are used. If no tab values are specified, single spaces replace tabs on output.
FA,NFA	File attributes. These codes specify whether or not the attributes (i.e., variable-length parameter list except name, account, and password) of the source file are to be carried over to the destination file. If the file name remains the same from source to destination and neither FA nor NFA is specified, the attributes are copied. If the names of the source and destination files are different, the attributes are not normally copied; information specified in ASSIGN or SET commands takes effect.
DEOD	Double end-of-file. Multiple source files are copied into a single output file. Thus, while COPY FT copies files including single end-of-file marks up to a double end-of-file, COPY FT (DEOD) copies files to a double end-of-file without copying the single end-of-file marks.
K	Reconstruct edit keys. If the file has a 3-byte key, the listing is not to be in hexadecimal form and the destination is a printer or terminal; the file is assumed to be an Edit format file. The use of the K option on output causes the key to be decoded as an Edit line number in the form xxxx.xxx and to be printed on the same line with the record contents (Edit or EDCON listing format). A record sequence number precedes the key. If the file is not an Edit format file, only the record sequence number precedes the record contents.

Examples:

1. Assume that file A is to be copied to labeled tape No. 4 with exactly the same attributes it had on RAD storage.

```
COPY A TO LT#4/A (RET)
..COPYING
```

2. Assume that RAD storage file A is in compressed form and is to be converted to symbolic and listed on the printer with double spacing.

```
COPY A (C) TO LP(DSP) (RET)
..COPYING
```

3. Assume that line images are to be read from RAD storage file A, converted from EBCDIC to Hollerith, and written on a 7-track scratch tape in BIN mode.

```
COPY DC/A TO FT(BIN, 7T, H) (RET)
..COPYING
```

4. Assume that a source file, SOURCE, containing tab characters was created on-line and is to be punched with tab characters expanded and carriage return characters removed.

```
COPY SOURCE TO CP(TX, NC) (RET)
..COPYING
```

RECORD SEQUENCING

Insertion or deletion of sequence identification for output data records is accomplished by using record sequencing specifications (Table 16). These specifications are available only as output options.

Examples:

1. Assume that a file called SORC on labeled tape #25 is to be sequenced and punched into cards. The card identification is SRCE, the initial value is 1, and the increment is 1. Thus, logical records are to be given sequential identification as follows: SRCE0001, SRCE0002, SRCE0003, etc.

```
COPY LT#25/SORC TO CP (CS(SRCE, 1, 1)) (RET)
..COPYING
```

2. Assume that PCL is to read successive records from free form tape #73, to assign line numbers starting at 5, in increments of 5, and to write the records on RAD storage file A.

```
COPY FT#73 TO DC/A(LN(5, 5)) (RET)
..COPYING
```

3. Assume that two keyed files A and B, are to be merged into file C and assigned new keys. Default keys are to be assigned.

```
COPY A, B TO C(LN) (RET)
..COPYING
```

Table 16. Record Sequencing Options – COPY Command

Code	Description
CS [(id[,n,k])]	Card sequencing in columns 73-80. id is identification (0-4 characters) n is initial value k is increment The identification (id) is left-justified in the field (73-80) and is followed by the sequence number, which is right-justified in the same field. Precedence is given to the sequence number if overlapping occurs. The default values for id, n, and k are null, 0, and 1, respectively.
NCS	No card sequencing. This specification strips columns 73-80 from each output data record.
LN[(n,k)]	Line numbering. The file starts at n and continues in sequential steps of k. Line number and increment formats are as in the Edit subsystem. Line numbers must be between 1 and 9999. Increments may range from .001 through 100.000. The default values for both n and k are 1.
NLN	No line numbering.

ASSIGNMENT OF ACCOUNTS

A maximum of eight read accounts and eight write accounts may be added as attributes of the output file as shown in Table 17.

Table 17. Account Options – COPY Command

Code	Description
RD(ac_1[,ac_2,...])	Adds read account(s) on output. A maximum of eight accounts may be given.
WR(ac_1[,ac_2,...])	Adds write account(s) on output. A maximum of eight accounts may be given.

Examples:

1. Assume that file A is to be copied to labeled tape No. 4 with the same attributes it had on RAD storage plus the addition of read accounts ONE and TWO.

```
COPY A TO LT#4/A(RD(ONE, TWO)) ⏎
..COPYING
```

2. Assume that read account ALPHA and write accounts X and Y are to be added as attributes of file SRCE.

```
COPY SRCE OVER SRCE(RD(ALPHA),WR(X,Y)) ⏎
..COPYING
```

RECORD SELECTION

This specification permits selection of the logical records to be copied by giving the sequential position of the records within the file. The specification has the form

X -Y

All records within the file that have a position, n, satisfying the condition X n Y are selected. Multiple selections may be specified if separated by commas (X-Y, U-V, W-Z). Selections do not have to be in sequential order (but non-sequential selection is very slow for tape operations). The maximum number of selections is ten for each input file.

Example:

Assume that sections of two files, N1 and N2, are to be combined to form a third file, N3. Records 20-30 and 40-100 of N1 followed by records 50-75 of N2 are to be copied, in that order, to N3. The job account is assumed for files N1 and N3; N2 is from account 34 under password PA.

```
COPY DC/N1(20-30, 40-100), N2.34.PA(50-75)
  TO DC/N3 ⏎
..COPYING
```

VALID OPTION COMBINATIONS

Not all combinations of source and destination devices, data types, formats, modes, or sequencing codes are valid. Table 18 shows the valid combinations, the invalid combinations, and the default provisions for the various possible combinations. If an invalid combination is found, an error message is produced. Execution of the command may or may not continue, depending on the severity of the error encountered (see Error Messages).

EXTENSIONS USING ASSIGN OR SET

Not all of the facilities available in the UTS I/O system are made available through PCL. More complicated data transfers may be specified by ASSIGN cards (batch mode) or SET commands (on-line mode). Since PCL reads through M:EI and writes through M:EO DCBs, special information, such as lists of read and write account numbers, may be prespecified by assigning either the input or output DCB.

ACCOUNT COPY COMMAND

This command allows all files, or a specified subset of files, in the log-on or some other account to be copied from a

Table 18. Valid Option Combinations

Option	Codes	Source Device							Destination Device							
		C	P	D	L	D	F	M	D	L	D	F	M	L	C	P
		R	R	C	T	P	T	E	C	T	P	T	E	P	P	P
Data codes	E	d	x	d	d	d	d	d	d	d	d	d	d	d	d	x
	H	x	-	x	x	x	x	-	x	x	x	x	-	-	x	-
Data formats	X	-	-	-	-	-	-	-	-	-	-	-	x	x	-	-
	C	x	x	x	x	x	x	-	x	x	x	x	-	-	x	x
Modes	None	-	d	d	-	d	-	d	d	-	d	-	-	-	-	d
	BCD	d	-	-	-	-	x	-	-	-	-	x	-	-	d	-
	BIN	x	-	-	d	-	d	-	-	d	-	d	-	-	x	-
	7T	-	-	-	x	-	x	-	-	x	-	x	-	-	-	-
	9T	-	-	-	d	-	d	-	-	d	-	d	-	-	-	-
	PK	-	-	-	x	-	x	-	-	x	-	x	-	-	-	-
	UPK	-	-	-	-	-	x	-	-	-	-	x	-	-	-	-
	SSP	-	-	-	-	-	-	-	-	-	-	-	d	d	-	-
	DSP	-	-	-	-	-	-	-	-	-	-	-	x	x	-	-
	VFC	-	-	-	-	-	-	-	-	-	-	-	x	x	-	-
	NC	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x
	CR	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x
	K	-	-	-	-	-	-	-	-	-	-	-	x	x	-	-
	FA	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-
	NFA	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-
	TX	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x
	DEOD	-	-	-	-	-	x	-	-	-	-	-	-	-	-	-
Sequencing	None	-	-	-	-	-	-	-	d	d	d	d	d	d	d	d
	CS	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x
	NCS	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x
	LN	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-
	NLN	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-
Accounts	RD	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-
	WR	-	-	-	-	-	-	-	x	x	x	-	-	-	-	-
Selection	x-y	x	x	x	x	x	x	-	-	-	-	-	-	-	-	-

Legend: d = default
x = optional
- = error, not available, unreasonable

file-type device (RAD, labeled tape, or disk pack) to any valid output device. It has the general form

COPYALL files [TO device]

where

files may be one of the following:

[DC][.acct][(s)][/r]
LT[#reel-id][(s)][/r]
DP[#reel-id][(s)][/r]

device may be one of the following:

DC[(a)]
LT[#serial no.][(a)]
DP[#serial no.][(a)]
FT[#serial no.][(7T)]
LP
ME
CP

In the above specification,

s may be KEY to copy keyed files only; or SEQ to copy sequential files only; or RAN to copy random files only; and/or 7T to copy from a 7-track tape; and/or PHY to copy in physical order from tape.

r may be f,t; or f; or ,t.

where

f is 1 to 31 characters representing the beginning of a range of files to be copied.

t is 1 to 31 characters representing the end of a range of files to be copied.

Both f and t are used as sort keys only and generally do not have to be file names. They may be written in character string or hexadecimal notation (e.g., A, 'A', or X'C1' all represent A.). The t field must be equal to or greater than the f field. Files on tape are assumed to be in alphanumeric order unless the PHY option is used.

If PHY is specified, the f and t fields define a physical range of files on tape instead of an alphanumeric range and therefore must be file names. If the f field is null, copying begins wherever the tape is positioned. If the t field is null, copying continues to end of tape. If the file in the f field does not exist, the command is aborted. If the file in the t field does not exist, copying continues to end of tape.

Note: The introductory slash (/) is optional if no codes or options precede it.

a may be RD with one to eight account numbers enclosed in parentheses, e.g., RD (XX, YY); and/or WR with one to eight account numbers enclosed in parentheses as for RD; and/or 7T to copy to a 7-track tape.

PCL copies all files from the input device to the output device. Files protected by passwords cannot be copied with this command unless the correct password is placed in the M:EI DCB by a SET command or an ASSIGN card. The BREAK key terminates execution of this command and causes PCL to type the identification of the last file copied.

A synonym file is copied to RAD or disk pack only if the parent file was copied or previously existed on the destination device. A synonym file is always copied to tape regardless of whether the parent file is present on the tape. If a range is specified on the command, the synonym files within the range are copied if the above conditions are met. A parent file of a synonym file within the range is not copied unless it is also within the range. If files are copied by organization (KEY, SEQ, or RAN option), synonym files are not copied.

If files are being copied to the terminal or line printer, each file copy is preceded by the name of the file. If files are being copied to any device other than the terminal, the message

..COPYING

prints when the first file copy begins.

If there are no files present in the specified account, the following message prints:

NO FILES IN DIRECTORY

If a file cannot be opened due to a password requirement, the following message is printed and execution of the command continues:

CAN NOT ACCESS FILE XXX

PCL indicates completion of the command by printing a message of the form

..nnnn FILES COPIED

where nnnn is the number of files copied during execution of the command.

Examples:

1. Assume that all files listed in the user's account directory are to be copied to labeled tape Nos. 3 and 4. Tape No. 4 is to be used only if No. 3 overflows.

```
<COPYALL TO LT#3#4 (RET)
..COPYING
```

Note that RAD storage space previously occupied by this account can be released for other use after the files have been copied.

2. Assume that files are to be restored on RAD storage under the job account from labeled tape Nos. 3 and 4, created under account :SYSGEN.

```
<COPYALL LT#3#4. :SYSGEN (RET)
..COPYING
```

3. Assume that an exact copy of labeled tape No. 3 is to be written on tape No. 4. The record size must fit the allowable installation-set allocation of core to a single job.

```
<COPYALL LT#3 TO LT#4 (RET)
..COPYING
```

4. Assume that all keyed files on disk pack #5 are to be written to a scratch tape.

```
<COPYALL DP#5 (KEY) TO LT (RET)
OUTPUT SERIAL NUMBER = XXXX
..COPYING
```

5. Assume that all files on RAD between the sort keys C and L are to be copied to the line printer. Each file name will print before the file copy. It is assumed that records are in BCD format.

```
<COPYALL C,L TO LP ®
...COPYING
```

6. Assume that all files on RAD are to have read accounts 123 and X'00C6' and write account XY added as attributes.

```
<COPYALL TO DC(RD(123,X'00C6'),
    WR(XY)) ®
..COPYING
```

CONTROL FILE COPY COMMAND

The control file copy command allows the copying of files whose identifiers appear in a control file. The command is called "copy standard" and has the form

COPYSTD input [TO output]

where

input may be one of the following:

[DC/]fid

LT [#serial no.] [(7T)]/fid

DP [#serial no.]/fid

output may be one of the following:

DC

LT [#serial no.] [(7T)]

DP [#serial no.]

FT [#serial no.] [(7T)]

LP

ME

CP

PCL opens the file named in the input specification and unless this file is specified as existing in the user's RAD account and the output device is 'DC', the file will be copied to the specified output device. The files named in the standard file are copied to the output device using the running account and the same file names as appear in the standard file for output.

The format of a standard file record is an initial character followed by name, account, and password separated by periods. For example:

```
*NAME.ACCT.PASS
*NAME.ACCT
*NAME
```

The initial character is unused in the copy operation. If no account is specified, then the source account for the file is assumed to be the same as the account of the standard file itself. Commentary may appear on each record.

Files named within the standard file may be from labeled tape, disk pack, or RAD; in fact all variations allowed for the input specification field of a COPY command are valid for these devices except that options are not allowed. Device codes and accounts present in the record override the one present on the COPYSTD command.

If files are being copied to the terminal or line printer, each file copy is preceded by the name of the file. If files are being copied to any device other than the terminal, the message

..COPYING

prints when the first file copy begins.

If a file does not exist or can not be opened due to a password requirement, the following message prints:

CAN NOT FIND OR ACCESS FILE XXX

The file is then bypassed and execution of the command continues.

The BREAK key terminates execution of the COPYSTD command and causes PCL to type the identification of the last file copied.

PCL indicates completion of the COPYSTD command by printing a message of the form

..nnnn FILES COPIED

where nnnn is the number of the files copied during execution of the command including the standard file itself.

Examples:

1. Assume that all files listed in file STDF on labeled tape No. 5 are to be copied to RAD storage. The format of file STDF is

```
*A    COMMENTARY
*B
*C
```

The command to be used is

```
<COPYSTD LT#5/STDF ®
..COPYING
```

On completion of the command, the files STDF, A, B, and C, will have been copied from tape No. 5 to the user's RAD account.

2. Assume that all files listed in file ST in the user's RAD account are to be copied to his account. The files named in ST must not currently exist in his account. The format of file ST is

```
.ALPHA.ACCT.PASS,BETA.:SYSGEN
:LT#5/B,C
```

The command to be used is

```
< COPYSTD ST (RET)
..COPYING
```

On completion of the command, only four files will have been copied: ALPHA and BETA from other accounts to the user's account, and files B and C from labeled tape No. 5.

3. Assume that all files listed in file :STD in account :SYSGEN are to be copied to the line printer. The files listed are all in account :SYSGEN. The format of file :STD is

```
=ALPHA,BETA,GAMMA
```

The command to be used is

```
< COPYSTD :STD. :SYSGEN TO LP (RET)
..COPYING
```

On completion of the command, files :STD, ALPHA, BETA, and GAMMA will have been copied from account :SYSGEN to the printer.

OTHER COMMANDS

This group of commands provides file deletion, file positioning, and other manipulation and maintenance functions.

DELETE The DELETE command deletes complete files and has the form

```
DELETE fid[,fid...]
```

where fid specifies the identification of the file to be deleted.

Example:

Assume that RAD storage file SOURCE is to be deleted. This file is assumed to have been set up under the log-on account with password PLEASE.

```
< DELETE SOURCE..PLEASE (RET)
..     1 FILES DELETED
```

Depressing the BREAK key terminates execution of the command. The summary message tells how many files were deleted.

DELETEALL Another delete command deletes all files, or a specified range of files, in the log-on account. The form of the command is

```
DELETEALL [from][,to]
```

where 'from' and 'to' are sort keys of 1 to 31 characters each that define a range of files to be deleted. Absence of a 'from' field indicates that files are to be deleted from the beginning of the account. Absence of a 'to' field indicates that files are to be deleted through the end of the account. Absence of both the 'from' and 'to' fields indicates that all files in the log-on account are to be deleted.

Both 'from' and 'to' are used as sort keys only and do not have to be file names. They may be written in character string or hexadecimal string notation (e.g., A, 'A', or X'C1' all represent A). The 'to' field must be equal to or greater than the 'from' field.

A synonym file within the range is deleted only if its parent file is within the range.

A confirmation, YES$, is required in the on-line mode. (This is shown in the examples below.)

If there are no files in the log-on account, PCL responds to the command with the following message:

```
NO FILES IN DIRECTORY
```

If a file cannot be opened due to a password requirement, the following message prints:

```
CAN NOT ACCESS FILE XXX
```

The file is then bypassed and execution of the command continues.

After the delete function is performed, the following message prints:

```
..nnnn FILES DELETED
```

The count (nnnn) does not include synonym files which were deleted.

Examples:

1. Assume that all files in the log-on account are to be deleted.

```
< DELETEALL (RET)
DELETEALL?
.YES$ (RET)
..     8 FILES DELETED
```

2. Assume that all files in the inclusive range B through H are to be deleted.

```
< DELETEALL B,H (RET)
DELETEALL?
. YES$ (RET)
.. 4 FILES DELETED
```

Depressing the BREAK key terminates execution of the command and causes PCL to type the identification of the last file deleted.

LIST The LIST command is of the form

```
        ⎡LT[#reel-id][(s)]
        ⎢ [DC[.acct]][(s)]
        ⎢LT[#serial no.][(s)]/fid[(s)][,fid[(s)]...]
LIST    ⎢fid[(s)][,fid[(s)]...]
        ⎢DP[#reel-id][(s)]
        ⎢DP[#serial no.]/fid[(s)][,fid[(s)]...]
        ⎣FT#serial no.[(s)]
```

All listed output goes through the M:LO DCB.

The action for the various specifications is as follows:

1. LT[#reel-id][(s)] (list file directory)

   Device options may be 7T, 9T, A, or EA

   PCL scans the labeled tape and lists the names of all files contained on it. If option A has been requested, the attributes of each file are also listed. These attributes include

   Size in granules.
   Record count.
   Organization (keyed or consecutive).
   Read accounts, if other than 'ALL'.
   Write accounts, if other than 'NONE'.
   Modification date.

   If option EA (extended attributes) has been requested, the following attributes are listed in addition to those described above:

   Creation date
   Expiration date
   Backup date
   Last access date

   If a file requires a password or account and none is given, this will be noted.

2. [DC[.acct]][(s)] (list file directory)

   Device option may be A or EA.

   PCL scans the user's RAD file directory and lists the names of all files. If A or EA has been specified, the attributes are listed as in 1.

3. LT[#serial no.][(s)]/fid[(s)][,fid[(s)]...] (list file attributes)

   This is a request for the attributes of the indicated files. Device options may be 7T or 9T. File options may be A (which is the default) or EA. If an account is required, it must be included in the file identifier. PCL prints an attribute summary for each file, as in 1.

4. fid[(s)][,fid[(s)]...] (list file attributes)

   This is a request for the attributes of the one or more RAD files named. Options may be A (which is the default) or EA. PCL prints an attribute summary for each file, as in 1.

5. DP[#reel-id][(s)] (list file directory)

   Device option may be A or EA.

   PCL scans the disk pack and lists the names of all files contained on it. If A or EA has been specified, the attributes are listed as in 1.

6. DP[#serial no.]/fid[(s)][,fid[(s)]...] (list file attributes)

   This is a request for the attributes of the indicated files on disk pack. File options may be A (which is the default) or EA. If an account is required, it must be included in the file identifier. PCL prints an attribute summary for each file, as in 1.

7. FT#serial no.[(s)]

   Device options allowed are 7T and 9T. Serial no. can be a fake. If the tape conforms to BPM labeling conventions, PCL prints the serial number, account, and contents (file names) of the tape. The tape remains positioned after the last file, thus enabling the user to add files.

If only the command LIST is given, and no specification follows, then the command executes as though it were LIST DC. LIST (A) and LIST.acct are also valid commands. All output, except for completion messages, is written through the M:LO DCB.

The BREAK key terminates execution of this command.

PCL indicates completion of the command by printing a message of the form

```
.. nnnn FILES LISTED
```

where nnnn is the number of files listed during execution of the command.

If attributes of all files in a RAD or disk pack directory are listed, the following message also prints:

```
..xxxx TOTAL GRANULES
```

Examples:

1. Assume that all files on RAD under the log-on account are to be listed.

```
< LIST ®
ALPHA
BETA
GAMMA
ZETA
.. 4 FILES LISTED
```

2. Assume that files on 7-track labeled tape Nos. 3 and 4 are to be listed. These tapes were created under the account :SYSGEN.

```
< LIST LT#3#4.:SYSGEN(7T) ®
SOURCE
ALPHA
XYZ
.. 3 FILES LISTED
```

3. Assume that the attributes of files ALPHA and BETA on RAD are to be listed. The attributes listed have the following meaning:

ORG	C = consecutive, K = keyed file, R = random file.
GRAN	Number of granules of RAD space (1 granule = 512 words).
REC	Number of records in file.
DATE	Modification date.
Name	File name.

Read and write accounts print on a separate line and will print only if they have other than default values.

```
< LIST ALPHA,BETA ®
ORG   GRAN   REC    DATE       NAME
 C      2     71   22 JUL 71   ALPHA
 K     14    590    1 AUG 71   BETA
.. 2 FILES LISTED
.. 16 TOTAL GRANULES
```

4. Assume that the extended attributes of file ABC on disk pack No. 2 are to be listed. This file has had write account 123 assigned previously.

```
< LIST DP#2/ABC(EA) ®
ORG   GRAN   REC    DATE       NAME
 C      28   385   16 AUG 71   ABC
  WRITE= 123
  WILL EXPIRE          31 DEC 71
  CREATED ON            2 AUG 71
  BACKED UP ON         10 AUG 71
  LAST ACCESS ON       18 AUG 71
.. 1 FILES LISTED
```

5. Assume that a type requires identification. The fake serial no. X is used in the command.

```
< LIST FT#X ®
INSN = 8522
ACCT = :SYSGEN
ONE
TWO
THREE
FOUR
FIVE
SIX
.. 6 FILES LISTED
```

REVIEW This command lists files in the log-on account and waits for a user response after listing each file name to allow the option of deleting the file. The format of the command is

REVIEW[from][,to]

where 'from' and 'to' are sort keys of 1 to 31 characters each which define a range of files to be reviewed. Absence of a 'from' field indicates that the account is to be reviewed from the beginning. Absence of a 'to' field indicates that the review is to continue to the end of the account. Absence of both the 'from' and 'to' fields indicates that the entire account is to be reviewed.

Both 'from' and 'to' are used as sort keys only and do not have to be file names. They may be written in character string or hexadecimal string notation (e.g., A, 'A', or X'C1' all represent A). The 'to' field must be equal to or greater than the 'from' field.

This command may be used in the batch mode and will function identically to 'LIST' except that a range specification is permitted.

The BREAK key terminates execution of this command.

Example:

```
≤ REVIEW N,X ®
--ENTER D TO DELETE FILE.
NAY ®
P .
W99 D *DELETED*
.. 3 FILES LISTED
```

Each file name within the inclusive range N through X is listed and a wait occurs. If a D is typed, the confirmation message *DELETED* prints, and the next file name is listed. If any character other than D is typed, including carriage return (®) or line feed (Ⓛ), the file is not deleted. Note that PCL responds immediately to the character that is typed (the period (.) and the D in the example above) and that a carriage return is not required. (The carriage return that occurred at the end of the line

```
P .
```

was provided by PCL.)

If a file has a password or is open by another user, this is noted by an appropriate message, and the review continues without the usual wait.

SPF This command positions free form tape forward or backward a designated number of files. The form of the command is

SPF FT[#serial no.][(7T)],±n

where

- \+ specifies forward direction.
- \- specifies backward direction.
- n is the number of files to be skipped.

If the direction is not given, forward direction is assumed. If an end-of-reel condition is encountered prior to completion, an error message is sent to the terminal.

Example:

Assume that free form tape No. 2076 is to be positioned forward two files.

≤ SPF FT#2076,+2 ®

SPE This command skips to the position following the last file on labeled tape. The form of the command is

SPE LT[#serial no.][(7T)]

Prior to issuing this command, the user must make sure that the tape is not write protected, i.e., the operator must be informed to insert a ring in the tape if it is a saved tape.

Example:

Assume that labeled tape No. 5 is to be positioned past the last file on the tape so that additional files may be added.

≤SPE LT#5 ®

WEOF WEOF writes an end-of-file on the current output device. This is an end-of-file mark for free form tape units, IEOD for card or paper tape punches, or top-of-form for line printers. The form of the command is

WEOF

(Note that only one output file will be open at a time.)

REW This command rewinds the specified magnetic tape reel. It has the form

REW[LT / FT] [#serial no.] [(7T)]

Example:

Assume that magnetic tape reel No. 205 is to be rewound.

≤REW#205 ®

REMOVE This command removes a magnetic tape or disk pack no longer needed, thus releasing the drive or spindle for other purposes. The form of the command is

REM[OVE][LT / FT / DP] [#serial no.] [(7T)]

If a tape is removed, the tape is rewound and a dismount message is sent to the computer operator. If a disk pack is removed, the user's interest in that spindle is released; however, no message is sent to the operator.

Example:

Assume that magnetic tape reel No. 2075 is to be rewound and removed.

≤REM#2075 ®

TABS This command sets tab values to be used in conjunction with the TX (tab expansion) option. As many as 16 values may be specified. The form of the command is

TABS s[,s]...[,s]

where s is a column position to be used in expanding a line.

Example:

Assume that tabs are to be set for standard Meta-Symbol list format.

≤TABS 10,19,37 ®

TERMINATION OF PCL

PCL operations are terminated by the END command. This command returns control to TEL.

Example:

```
< END (RET)
!
```

ERROR MESSAGES

PCL uses the ERRMSG file (see Appendix B). If there is an error message in the ERRMSG file, PCL sends that message to the terminal instead of the decimal code. If there is no error message in the file, PCL sends a decimal code.

PCL has two types of error conditions. One type consists of the I/O error and abnormal conditions as listed in Appendix B. The other type consists of errors arising out of the use of PCL commands. These conditions are defined in Table 19.

Table 19. PCL Error Codes

Decimal Code	Message	Severity Level
01	Argument greater than 31 characters.	2
02	Illegal device code.	2
03	More than four characters in a reel number specification.	2
04	Illegal file name specification.	2
05	Illegal account number specification.	2
06	Illegal password specification.	2
07	Too many fields in a file identification specification.	2
08	Invalid file range specification.	3
09	More than ten RS fields for an input device[t].	2
10	Overflow on an RS value.	2
11	Error on Y value of RS option[tt].	2
12	CS ID-field greater than four characters.	1
13	Error on N or K value of CS option.	1
14	Improper termination within RS, LN, or CS option.	3
15	)) must terminate RS, LN, or CS option.	3

[t] RS signifies record selection.

[tt] Y signifies the upper limit of a record selection.

Table 19. PCL Error Codes (cont.)

Decimal Code	Message	Severity Level
16	Special arguments must have) as termination character.	3
17	EH?	3
18	Undefined command action verb.	2
19	Illegal input device.	3
20	No defined output devices.	3
21	Illegal output device.	2
22	Reel number specification not valid for this device.	2
23	File specification not valid for this device.	2
24	Data code specification not valid for this device.	2
25	Mode specification not valid for this device.	2
26	Sequence specification not valid for this device.	2
27	Record selection specification not valid for this device.	2
28	PK/BIN/7T combination not valid.	2
29	Null arguments (two delimiters in a row).	1
30	Improper termination of the command.	1
31	One reel number must be specified on this command.	2
32	'TO' or 'OVER' not specified.	3
33	Record size exceeds available memory.	3
34	Invalid device type for this command.	3
35	More than three reel numbers specified.	3
36	Overflow on number of files on 'SPF' command.	3
37	Invalid direction indicator on 'SPF' command.	3
38	Input record size larger than 32767 bytes.	3
39	Invalid option for COPYALL.	3
40	Account specification not valid on 'SPE' command.	3

Table 19. PCL Error Codes (cont.)

Decimal Code	Message	Severity Level
41	RS specification beyond end of file.	2
42	Error in compressed input.	3
43	C option invalid on both input and output.	3
44	Record selection invalid with C option.	3
45	Invalid tab specification.	3
46	Overflow on Edit line number.	3
47	Zero increments on CS or LN option.	1
48	TX option used without TABS command.	1
49	Invalid option for COPYSTD.	2
50	More than eight read or write accounts.	1
51	More than 16 tab values.	1
52	Unable to dismount.	2

A severity level of 1, 2, or 3, is attached to each error and has the following effect on the execution of the command in question:

1. Warning

   PCL continues execution. The message will be printed only if a higher error severity level occurs during execution of a command.

2. Invalid Syntax or I/O Error

   This level terminates execution of the command but continues the syntax edit of the command for both on-line and batch operations.

3. Format Error

   This level terminates the command.

   In the case where a command is terminated (severity level 2 or 3), PCL reverts to the command state if the error occurs during on-line operations; it reads the next command card if the error occurs during batch operations.

Example:

Assume that a file is to be copied from RAD storage file A to the card punch. In entering the command, the device code for the RAD is entered as CC instead of DC.

<COPY CC/A TO CP (RET)

Error message printout:

ILLEGAL DEVICE CODE

PCL COMMAND SUMMARY

Table 20 is a summary of PCL commands. The left-hand column gives the command formats. The right-hand column gives the command function and options.

Table 20. PCL Command Summary

Command	Description
COPY d[(s)][/fid[(s)][,fid[(s)]]...][;d[(s)] [/fid[(s)]][,fid[(s)] ...]... [TO / OVER] d[(s)][/fid[(s)]]	Copies file between devices or between RAD storage and devices. Options: d may be CP, CR, DC, FT, LP, LT, or ME. s may be a data code (E, H); a data format (X,C); a mode (BCD, BIN, 7T, 9T, PK, UPK, SSP, DSP, VFC, NC, FA, NFA, TX, DEOD, K); a sequence (CS, NCS, LN, NLN); an account (RD,WR); or selection (x-y).
COPYALL {[DC][.acct][(s)][/r] / LT[#reel-id][(s)][/r] / DP[#reel-id][(s)][/r]} [TO d[(a)]]	Copies files from RAD, labeled tape, or disk pack to any output device. Options: d may be DC, LT, DP, FT, LP, ME, or CP. s may be KEY, SEQ, RAN, or 7T. r is a range specification. a may be RD, WR, or 7T.

Table 20. PCL Command Summary (cont.)

Command	Description
COPYSTD { [DC/]fid / LT[#serial no.] [(7T)]/fid / DP[#serial no.] /fid } [TO d[(7T)]]	Copies a control file and all files named within the file. Option: d may be DC, LT, DP, FT, LP, ME, or CP.
DELETE fid[, fid...]	Deletes the specified files.
DELETEALL [from][,to]	Deletes all files or a specified range of files in the log-on account and requires a confirmation: DELETEALL? _ YES$ (RET) ..nnnn FILES DELETED
END	Returns control of the terminal to TEL.
LIST [LT[#reel-id][(s)] / [DC[.acct]][(s)] / LT[#serial no.] [(s)]/fid[(s)][,fid[(s)]...] / fid[(s)][,fid[(s)]...] / DP[#reel-id][(s)] / DP[#serial no.]/fid[(s)][,fid[(s)]...] / FT #serial no. [(s)]]	Lists file names and, optionally, attributes from the account dictionary, tape, or disk pack. Option: s may be A, EA, 7T, or 9T.
REM[OVE] [LT / FT / DP] [#serial no.][(7T)]	Removes a magnetic tape or disk pack.
REVIEW [from][,to]	Reviews all or a portion of files in the log-on account.
REW [LT / FT] [#serial no.][(7T)]	Rewinds tape reel.
SPE LT[#serial no.] [(7T)]	Spaces to the end of the last file on labeled tape.
SPF FT[#serial no.][(7T)],±n	Positions free form tape forward or backward a designated number of files.
TABS s[,s]...[,s]	Sets tab values for tab expansion.
WEOF	Writes an end-of-file on the current output device.

6. EDIT

INTRODUCTION

Edit is a line-at-a-time context editor for on-line creation, modification, and manipulation of files of EBCDIC text. All Edit data is stored on disk in a keyed file structure of sequence-numbered variable-length records, which permits Edit to directly access each line or record of data. Edit functions are controlled via single-line commands from the user. The command language provides for the following:

1. Creating a sequenced EBCDIC coded text file.
2. Inserting, reordering, and replacing lines or groups of lines of text.
3. Selective printing and renumbering.
4. Reordering groups of records within a file.
5. Merging part of one file into another.
6. Context editing operations that allow matching, moving and substituting character strings within a specified range of text lines.
7. Maintaining files (allowing the user to build, copy, and delete whole files of text lines).

A user may edit files under his own account (i.e., the one under which he logged on) or under accounts to which he has been granted write-access by the file creator. He may copy his own files or those to which he has read-access. Under the rules of UTS file access, a file may not be created (i.e., built or copied to) under an account number different than that used for log-on.

In using Edit, it must be stressed that the edit takes place as the commands are given; the file is edited in place. Therefore, a backup file should be kept to protect against user or machine errors.

CALLING EDIT

An on-line user of UTS may call the Edit processor either directly,

```
!EDIT
```

or indirectly through one of two executive-level commands:

```
!EDIT fid      (edit an existing file)
!BUILD fid     (build new file)
```

The first executive-level command allows the user to call Edit for updating an existing file. Edit first opens the specified file and then prompts for command input by typing its identifying mark, the asterisk (*). The second executive-level command allows the user to call Edit for on-line creation of a text file. Edit opens the specified file and prompts for command input by typing the first line number at the left margin of a fresh line. The user is expected to enter the text lines of the new file.

If an Edit command is given at the executive level without a file identifier, Edit types EDIT HERE and prompts for further commands by typing an asterisk (*).

RECORD FORMATS

The editing process is based on a sequence number associated with each line. Unsequenced files of text lines may be sequenced via the Edit COPY command. Sequence numbers for inserting new lines may be generated automatically by Edit or may be supplied by the user.

Sequence numbers consist basically of an integer and three fractional digits. However, the user may write a sequence number with one or more fractional digits omitted and Edit will automatically assume sufficient trailing zeros to complete the sequence number. For example:

Sequence Number	Implies
50	50.000
50.01	50.010
50.5	50.500
50.008	50.008

Edit writes variable-length records, with a maximum record size of 140 characters including the ⏎. Trailing blank characters in a record are not written on the file.

Edit files are stored on disk as keyed records, with the keys being binary representations of the sequence numbers. The sequence number DDDD.DDD is taken as a seven-digit decimal integer and converted to binary, giving a key with a maximum length of three bytes. For example, the following record created in a BUILD operation would have a key value of 8000_{10} and a record length of 20 bytes (assuming that ⏎ is in column 20):

```
8.000    B2    LI,5    0 ⏎
```

If the ⏎ is preceded by a number of blanks, they will not be carried in the output. The record terminator can be either ⏎ or ␊ and is carried in the record as X'15'.

MULTILINE RECORDS

On a terminal unit having an inherent line-width limit of less than 140 (e.g., Teletype models 33, 35, and 37), a

single, multiline record may be entered into a file (using the BUILD or IN commands, for example) in either of two ways:

1. Using the local carriage return key marked LOC CR, if present, to "break" the input line without releasing it to the system.

2. Using the simulated local carriage return sequence (ESC) (RET) for the same purpose.

Either method permits entering a record of up to 139 characters plus (RET) on virtually any terminal unit.

An example of a multiline record is presented in Figure 3.

BREAK FUNCTION

The BREAK key always causes an immediate interruption in Edit activity, with any partially completed input being discarded and any waiting output being delivered to the terminal. Edit stops any command in progress and reverts to accepting command input from the user.

If the command in progress when an interrupt occurs is a display command (for example, TY), the display will stop within the next several lines after the interrupt is given.

For commands that produce no display while operating on a range of records, the point of interrupt is reported by a message which denotes the sequence number(s) of the record(s) being processed at the time of the interrupt. Edit then types this message

--ENTER X TO ABORT COMMAND. ANY OTHER CHARACTER CONTINUES.

and prompts for a single character input. If the user enters an X, the operation aborts; if he enters any other character, the operation continues.

If a command is being executed and the BREAK key causes an interrupt during an I/O operation (e.g., READ, WRITE, OPEN, DELETE record), the I/O operation is completed. After the I/O is completed, the user may continue execution of the command to normal conclusion or may immediately terminate the command. With record or intrarecord commands (see Command Structure), the current Edit file remains open. All file commands terminate by closing all files.

EDIT COMMANDS

COMMAND STRUCTURE

Edit commands fall into the following three categories:

1. File commands: Commands that apply to an entire file. These commands may be given at any time.

2. Record commands: Commands that act upon one record or a group of records within a file. These commands may be given only after a file has been selected for editing.

3. Intrarecord commands. Commands that make changes within an individual record. These generally manipulate character strings. These commands may be given only after a specific set of records has been selected by a command of type 2, above (either the SE, SS, or ST command.)

FILE COMMANDS

The file commands will be discussed in the following order:

EDIT	Select file for editing.
BUILD	Create a new file.
COPY	Copy file 1 to file 2.
DELETE	Delete file.
MERGE	Merge files.
END	Exit to executive.
CR	Set carriage return mode.
TA	Set tab positions.
BP	Set blank preservation mode.

Line number 4.000 is input as a multiline record in the following manner:

```
4.000 THIS IS AN EXAMPLE OF A MULTILINE(ESC)(RET)
RECORD. A RECORD CAN CONTAIN UP TO 140 (ESC)(RET)
CHARACTERS INCLUDING THE CARRIAGE RETURN.(RET)
```

If this record were displayed by Edit, it would appear as

```
4.000 THIS IS AN EXAMPLE OF A MULTILINERECORD. A RECORD CAN CONTAIN U
P TO 140 CHARACTERS INCLUDING THE CARRIAGE RETURN.
```

Note that the user did not type a space after the word 'multiline' and that Edit did not assume a space. Also, the system "folds" the record indiscriminately when the physical line width limit is reached.

Figure 3. A Multiline Record

EDIT Edit File

EDIT opens a file to be edited. The EDIT command has the format shown below.

*EDIT fid

The EDIT command must be used to enter the record editing mode and to identify the file that is to be edited.

Use of any of the following commands terminates the record editing mode: BUILD, DELETE, MERGE, and COPY. If an EDIT command is given while in the record editing mode, the previously open file is closed and the specified file is opened. In both situations, the following message is printed by Edit:

..EDIT STOPPED

Edit then processes the new command.

Edit responds to user errors with the following messages:

-NO SUCH FILE The file does not exist.

-FILE NOT KEYED; MUST COPY The file is not in the keyed format needed by Edit and must be copied (via the COPY command) before it can be edited.

BUILD Build New File

The BUILD command enables the user to create a new file. The command may be given at the executive level with the form

!BUILD fid

or it may be given at the Edit subsystem level with the form

*BUILD fid[,n[,i]]

where

fid is the identifier of the file to be created.

n is the sequence number at which the new file is to start. The default value is 1.

i is the value by which sequence numbers for the new file are to be incremented. The default value is 1.

The system prompts by typing a sequence number, and the user then types in the corresponding line. A null line (indicated by ⏎ alone) terminates the build operation and closes the file. If the BUILD command is used at the executive level, then control returns to the executive level after typing a null record (⏎ alone); and if the BUILD command is invoked while in the Edit subsystem, then control returns to the Edit subsystem after typing a null record.

Example:

```
*BUILD SOFILE ⏎
1.000          SYSTEM    SIG5 ⏎
2.000          DEF    B ⏎
3.000          REF    A ⏎
4.000          B    A ⏎
5.000          END ⏎
6.000 ⏎        The null record, consisting of
*              only a carriage return, termi-
-              nates the command and does not
               appear in the output file.
```

Edit responds to user errors with the following messages:

--OVERFLOW More than 140 nonblank characters were entered.

-FILE EXISTS:CAN'T BUILD A file with the same name (fid) already exists.

COPY Copy File

COPY causes Edit to copy a specified file. The COPY command has the format shown below.

$$\text{*COPY fid}_1 \left\{ \begin{matrix} \text{ON} \\ \text{OVER} \end{matrix} \right\} \text{fid}_2[,n[,i]]$$

where

fid_1 identifies the file that is to be copied.

fid_2 identifies the file to which fid_1 is to be copied.

n is the starting sequence number for the new file. If omitted, the old sequence numbers of fid_1 are retained in the copy.

i is the sequence number increment for the new file. The default value is 1.

If ON is specified, a new file is created (and must not already exist). If OVER is specified, fid_2 may exist; and if it does, it will be deleted and replaced by the copy of fid_1.

Example:

```
*COPY  PROG1  ON  PROG2 ⏎
..COPYING
..COPY DONE
```

Edit responds to user errors with the following messages:

-P2:FILE EXISTS A COPY ON has been given but fid_2 exists.

-P1:NO SUCH FILE The file identified by fid_1 does not exist.

-P1:FILE NOT KEYED & P3 NULL. There are no keys on the file identified by fid_2 and resequencing has not been specified; thus, if copied, the resultant file could not be edited.

DELETE Delete File

DELETE causes Edit to delete a specified file from the log-on account. The DELETE command has the format shown below.

*DELETE fid

Example:

*DELETE PROG1 (RET)

..DELETED The file has been deleted.

If the file does not exist, Edit prints the message:

-NO SUCH FILE

MERGE Merge Files

MERGE causes Edit to transfer records between specified files. The MERGE command has the form shown below.

*MERGE $fid_1[,n_1[-n_2]]$INTO $fid_2,n_3[-n_4][,i]$

Records n_1 through n_2 from file fid_1 are merged into file fid_2 where they replace records n_3 through n_4. In the target file the new records are numbered from n_3 in steps of i. The source file, fid_1, must be keyed format or else Edit aborts the command. If no range specification is attached to fid_1, all of its records are subject to the move. If a range specification exists, Edit checks that at least one record is contained in it.

Example:

*MERGE	fid_1	INTO	Merges all of fid_1.
*MERGE	fid_1,10	INTO	Merges record 10.000 of fid_1.
*MERGE	fid_1,10-12.5	INTO	Merges records 10.000 through 12.500.

After validity checks are made on fid_1, Edit checks for the existence of fid_2. If fid_2 does not exist, Edit creates a file identified by fid_2 and then moves the appropriate record set from fid_1 into fid_2, resequencing from n_3 and incrementing by i. (If no value for i is specified, the value is 1 by default.) This operation is similar to a COPY operation, except for the selection of records from fid_1. If fid_2 exists, Edit deletes from it all records in the range n_3-n_4 and then replaces them with the appropriate records from fid_1, starting at sequence n_3 and incrementing by i.

Example:

*MERGE ALPHA.ACCT1,100-120 INTO BETA, 400-440 (RET)

..MERGE STARTED

--DONE AT 420 420 is the last sequence number assigned in BETA.

If (when fid_2 exists) the number of records to be transferred at the specified increment causes Edit to equal or exceed the next higher existing sequence number above the destination range n_3-n_4, the merge is stopped with the message

--CUTOFF AT $n_5(n_6)$

where

n_5 is the last sequence number assigned in fid_2.

n_6 is the sequence number of the last record moved from fid_1.

The user may then give subsequent commands to investigate how to move the remaining records.

Edit responds to user errors with the following messages:

--EOF HIT The range of n_3-n_4 passes beyond the end-of-file in fid_2.

-P1:NO SUCH FILE The parameter fid_1 does not exist.

-NOTHING TO MOVE The specified range in fid_1 contains no records.

-MERGE SOURCE NOT KEYED The parameter fid_1 must be a keyed file.

-MERGE DESTINATION NOT KEYED The parameter fid_2 must be a keyed file.

END Exit

END causes the Edit to close all active files and return control to the terminal executive language (TEL). The END command has the format:

*END

Example:

*END (RET)

! Any TEL command may now be given.

CR Set Carriage Return Mode

The CR command controls the inclusion of the CR (X'15') character at the end of each record in the user's output file. The CR command has the form shown below.

*CR {ON / OFF}

where

ON includes the X'15' terminator in the user's output file.

OFF excludes the X'15' terminator from the user's output file and is the default setting.

The carriage return is normally not included since this is provided by the COC routines. However, if the user wishes to reproduce the file on cards or tape (for later use by other than UTS software), he may want the carriage return. Inclusion of the carriage return character will have no effect on the typing of records on the Teletype, however.

The CR command may be given at any time.

If a parameter other than ON or OFF is specified, Edit prints the message:

-NOT ON/OFF

TA Set Tab Positions

TA causes the Editor to set or reset the terminal tab stops.

*TA {F / M / S}

where

F implies FORTRAN and a tab set at column 7.

M implies Meta-Symbol and tabs set at columns 10, 19, and 37.

S implies Meta-Symbol, short-form, and tabs set at columns 8, 16, and 30.

These tab settings correspond to record column numbers and are offset to provide for the line number produced at the left margin of the user terminal. The TA command may be given while an edit operation is in progress without stopping the edit operation, but it may not be used as an intrarecord command.

If the parameter supplied is not from the legal set, Edit prints the message:

-NOT F/M/S

When the programmer uses the Teletype to build a file, he can columnarize the instructions as if he were typing them on a coding sheet. However, unlike the TAB key on most typewriters, the TAB key on many Teletypes does not move the carriage across the page. Therefore, a UTS service is provided to simulate tabbing action when the TAB key is struck. To achieve simulation the user must do three things:

1. Tell the system where the tab stops are by using the executive command TABS or the Edit command TA.
2. Be sure tab simulation is on to cause the appropriate number of spaces to be sent on output and echoed on input whenever a tab character is detected. (Tab simulation is discussed in Chapter 10.)
3. Set space insertion mode. If space insertion mode is on, an appropriate number of spaces will be inserted into the input record. If space insertion mode is off, the tab character (X'05') will be inseted into the input record. (Space insertion mode is discussed in Chapter 2.)

Edit puts the actual tab character (X'05') into the file being constructed whenever the TAB key is struck, regardless of whether simulation is carried out.

When using intraline commands to edit text that contains tab characters, the user must give a TA or TABS command so that Edit will know how to interpret the tab characters it finds. Edit then uses this information to expand the records by inserting an appropriate number of blanks for each tab character it finds. (See the discussion of the blank preservation command, BP, later under "Intrarecord Editing Commands".) If tab stops have not been set by a TA or TABS command and Edit finds a tab character, the user is notified with the message

-TAB CHARACTER FOUND. NO TAB STOPS SET.

BP Set Blank Preservation Mode

BP sets the blank preservation mode on or off. The BP command has the format shown below.

*BP [{ON / OFF}]

When "on", all strings of blanks are preserved during intrarecord operations. When "off", blank strings are compressed to a single blank or expanded as required to retain column alignment of nonblank fields. The default mode is "off".

When a string is inserted or replaced in a manner that changes the number of characters in a record, the record format is adjusted as follows.

When the blank preservation mode is off, the blanks between two successive strings are not preserved. When a string operation causes the first of two strings to be expanded or contracted, the number of blanks between the two strings are decreased or increased so that the second string stays in the same columnar position. (If the first string expands, the number of blanks between the two strings decreases; if the

first string contracts, the number of blanks increases.) At least one blank must be left between strings.

When the blank preservation mode is on, the blanks between the two strings are preserved. That is, when the first string expands or contracts, the second string is moved to the left or right so that the same number of blanks remains between the two strings.

For example, the following string substitution command

```
*/8/S/LINK/(RET)
```

substitutes the string "LINK" for the string "8" in the instruction

```
$10 BAL,8   SUB
```

adjusting blanks as indicated below:

old	$10BAL,8 SUB
new (BP-OFF)	$10BAL, LINK SUB
new (BP-ON)	$10BAL, LINK SUB

Although the BP command is discussed with the file commands, it may also be used when Edit is in the record mode or the intrarecord mode.

If a parameter other than ON or OFF is specified, Edit prints this message:

```
-NOT ON/OFF
```

RECORD EDITING COMMANDS

The record editing commands may only be given after a file has been opened for editing via the EDIT command. If the user does not open a file for editing before giving a record editing command, Edit prints the message:

```
-NO FILE NAMED
```

The record editing commands will be discussed in the following order:

IN, IS	Insert records.
DE	Delete records.
TY, TC, TS	Type individual records.
MD, MK	Reorder records within a file.
FD	Delete records containing a specified character string.
FT	List sequence numbers and contents of records containing a specified character string.
FS	List sequence numbers of records containing a specified character string.
RN	Renumber record.
CM	Insert commentary.
SE	Select a group of records for character operations.
SS, ST	Select records for step mode operation.

IN Insert New Records

IN causes Edit to insert new records into a file. The IN command has the format shown below.

```
*IN n[,i]
```

New records are inserted starting at the record with sequence number n, with each successive record being sequenced from n with increment i. (If i is omitted, the increment size specified in the most recent record editing command is used. If no such commands have been given, the value 1 is assumed by default.) If a record with sequence number n exists in the file, it is replaced by the newly inserted record n.

Edit prompts the user console with the first sequence to be inserted, and repeats the prompt for each subsequent insertion, increasing the sequence number by the increment i.

The insertion can be terminated in either of two ways. If a null record ((RET) only) is supplied, the insertion terminates. An equivalent action takes place if an incremented sequence equals or exceeds a sequence existing in file. In the latter case, the console bell is rung.

Example:

```
*EDIT SOURCEFILE (RET)
*IN 100, .1 (RET)
100.000    10 A = 2.5(RET)   Replaces the existing record.
100.100       B = 0.(RET)
*                            Record insertion terminates
                             because sequence number
                             100.200 existed previously;
                             the console bell is rung.
```

If the user types more than 140 nonblank characters, Edit prints the message

```
--OVERFLOW
```

IS Insert New Records

The IS command is identical to the IN command in function and format except that Edit does not prompt with sequence numbers. The format of the IS command is:

```
*IS n[,i]
```

Example:

```
*EDIT SOMEFILE (RET)
*IS  100, .1 (RET)
10  A = 2.5 (RET)
    B = 0 (RET)
(RET)

*
```

If the user types more than 140 nonblank characters, Edit prints the message

```
--OVERFLOW
```

DE Delete Records

DE causes Edit to delete all records whose sequence numbers lie in a specified range. The DE command has the form shown below.

```
*DE n[-m]
```

where

n specifies the number of the first record to be deleted.

m specifies the number of the last record to be deleted. If m is omitted, only record n is deleted.

Example:

```
*DE 50 (RET)          Deletes record 50.000 only.

*DE 50-60.5 (RET)     Deletes all records in the range
                      50.000 through 60.500, inclusive.
```

If no records are found in the specified range, Edit prints the message:

```
--NOTHING TO DE
```

If the range n-m passes beyond the end-of-file, Edit prints the message:

```
--EOF HIT
```

TY Type Records

TY causes Edit to type the sequence numbers and the contents of specified columns of one or more records. The TY command has the format shown below.

```
*TY n[-m][,c[,d]]
```

Edit types records in the range n to m, and types only the portions between columns c and d. If m is omitted, only record n is typed. If the values for c and d are not given, c has a value of 1 and d has a value of 140 by default.

Example:

```
*EDIT SOURCEFILE (RET)
*TY 1-2,4,8 (RET)
1.000   EQU
1.200   SYST
1.400   REF
1.600   DEF
1.800   PAGE
2.000   ITIAL
*
```

Edit responds to user errors with the following messages:

-BAD COL.NO. PAIR The columns specified are not in the range 1 through 140, or c > d.

--EOF HIT The range n-m passes beyond the end-of-file.

TC Type Compressed

TC causes Edit to type the sequence numbers and the contents of specified columns of one or more records. Any nonblank strings within the columns are shifted to the left to compress each blank string to a single blank. This compression affects only the typed output; the records themselves are not affected. TC is the same as TY with all blank strings compressed to a length of one. The TC command has the format:

```
*TC [n -m][,c[,d]]
```

Edit types records in the range n to m, and types only the portions between columns c and d. If m is omitted, only record n is typed. If the values for c and d are not given, c has a value of 1 and d has a value of 140 by default.

Example:

```
*EDIT SOURCEFILE (RET)
*TC 1-2,1,7 (RET)
1.000 A EQU
1.200  SYS
1.400 B REF
1.600 C DEF
1.800 PAGE
2.000 *INITIA
*
```

Edit responds to user errors with the following messages:

-BAD COL.NO. PAIR The columns specified are not in the range 1 through 140, or c > d.

-EOF HIT The range n-m passes beyond the end-of-file.

TS Type Records, Suppressing Sequence Number

TS causes Edit to type the contents of specified columns of one or more records, without accompanying sequence numbers. The TS command has the format shown below:

*TS [n -m][,c[,d]]

Edit types records in the range n to m, and types only the portions between columns c and d. If m is omitted, only record n is typed. If the values for c and d are not given, c has a value of 1 and d has a value of 140 by default.

Example:

```
*EDIT SOURCEFILE (RET)
*TS 1-2,1,8 (RET)
A    EQU
     SYS
B    REF
C    DEF
     PAG
*INITIA
*
```

Edit responds to user errors with the following messages:

-BAD COL.NO.PAIR The columns specified are not in the range 1 through 140, or c > d.

--EOF HIT The range n-m passes beyond end-of-file.

MD Move and Delete Records

MD causes Edit to move records from one specified range to another. Records originally in the destination range are deleted. The MD command has the form shown below.

*MD [n -m],k[-p][,i]

where

n specifies the sequence number of the first record that is to be moved.

m specifies the sequence number of the last record that is to be moved. If omitted, only n is moved.

k specifies the lower limit (i.e., sequence number) of the range of destination records that will be deleted.

p specifies the upper limit of the range of records to be deleted. If omitted, only k is deleted. However, records from the range n-m are still moved to record k and following.

i specifies the increment value to be used for renumbering records. If omitted, the most recent increment value specified in a record edit command is used. If no such commands have been given, the default value is 1.

The first record (n) is renumbered as k. Successive records from the range n-m are renumbered consecutively higher, incremented by i.

It is important to note that the ranges n-m and k-p may not overlap.

As each record from the range n-m is moved, it is deleted from the original range (n-m). At the end of this operation, a message is printed specifying the new sequence number of the last record moved from the range n-m.

Example:

```
*EDIT BETA (RET)
*MD 5-21, 100-101, .02 (RET)
--DONE AT 100.32
```

If the increment is too large to permit all records in the range n-m to be moved into the space between k and the next record after p, a message is printed specifying the sequence numbers, from both ranges, of the last record moved.

In this case the original contents of range k-p will be lost, but only those records in the range n-m that have actually been moved will have been deleted. Thus, the user can perform another move (with a smaller increment) to move the remaining records in the range n-m.

Example:

```
*EDIT BETA (RET)
*MD 10-30, 100-110, 1 (RET)
--CUTOFF AT 110. (20.)
```

20 is the number of the last record that was moved.

Edit responds to user errors with the following messages:

--NOTHING TO MOVE No records exist in the range n-m.

--RNG OVERLAP Ranges n-m and k-p overlap.

--EOF HIT Range n-m passes beyond the end-of-file.

MK Move and Keep Records

MK is identical to MD except that the records in the range n-m are not deleted as they are moved; thus a copy of records in the range n-m is made. The MK command has the form shown below.

*MK [n -m],k[-p][,i]

FD Find and Delete Records

FD causes Edit to search for a specified string between specified columns. If the string is found, the record containing it is deleted from the file. The FD command has the form shown below.

*FD [n -m],/string/[,c[,d]]

where

n specifies the sequence number of the first record to be searched.

m specifies the sequence number of the last record to be searched. If omitted, only record n is searched.

/string/ specifies the character string identifying the record to be deleted.

c specifies the lower limit (i.e., column number) of the field to be searched. The default value is 1.

d specifies the upper limit of the field to be searched. The default value is 140.

The specified string must be entirely contained within columns c through d to cause deletion. At the end of this operation, a message is printed telling how many records were deleted.

*EDIT FILEA ⏎
*FD 5-20.4,/DATA/, 10, 18 ⏎
--006 RECS DLTED

If there are no records in the specified range containing the indicated string, Edit prints the following message:

--NONE

Edit responds to user errors with the following messages:

-BAD COL. NO. PAIR The columns specified are not in the range 1 through 140, or c > d.

--EOF HIT The specified range passes beyond the end-of-file.

FT Find and Type

FT causes Edit to search for a specified string between specified columns. If the string is found, Edit types out the sequence number and the contents of the record. (The string must be entirely contained within the specified columns.) The FT command has the format:

*FT [n -m],/string/[,c[,d]]

The parameter specifications are the same as those for the FD command.

Example:

*EDIT SOMEFILE ⏎
*FT 1-100,/LW/,10 ⏎

5.000	LW,3	DATA
9.000	LW,2	TABLE,7
21.480	LW,10	LOC+5,8
73.000	LW,9	FLAG

*

If there are no records in the specified range containing the indicated string, Edit prints the message

--NONE

Edit responds to user errors with the following messages:

-BAD COL. NO. PAIR The columns specified are not in the range 1 through 140, or c > d.

--EOF HIT The specified range passes beyond the end-of-file.

FS Find String and Type Sequence Number

FS causes Edit to search a given range of records for a specified character string between designated columns. Edit will type the sequence number of each record satisfying the search criteria. The FS command has the format:

*FS n[-m],/string/[,c[,d]]

The parameter specifications are the same as those for the FD command.

Example:

*EDIT SOMEFILE ⏎
*FS 10-20,/BE/, 10, 11 ⏎
15.000
18.000
*

If there are no records in the specified range containing the indicated string, Edit prints the following message:

--NONE

Edit responds to user errors with the following messages:

-BAD COL. NO. PAIR The columns specified are not in the range 1 through 140, or c > d.

--EOF HIT The specified range passes beyond the end-of-file.

RN Renumber Record

RN causes Edit to renumber a specified record. The RN command has the form shown below.

*RN n,k

This has the same effect as deleting record n and then entering a new record with sequence number k with the same contents as n. Sequence number k must not already exist.

Edit responds to user errors with the following messages:

-P1:NO SUCH REC Record n does not exist.

-P2:REC EXISTS Record k already exists.

CM Commentary

CM causes Edit to insert commentary into specified columns of each successive record beginning at a specified sequence number. The CM command has the format shown below.

*CM n,c

where

n is the record number.

c is the column number.

The sequence number of each record is typed and then the user types in the data he wants inserted starting at column c. The data he types in is blank filled to the right through column 140, as required. A null record terminates the command. It is not necessary to delimit commentary with slashes.

Example:

*EDIT SOURCEFILE (RET)

*CM 37.6, 40 (RET)

37.600 * COMMENT 1 (RET)

37.800 * COMMENT 2 (RET)

40.500 * (RET)

*

Edit responds to user errors with the following messages:

-P2:COL ERROR Column c > 140.

-P1:NO SUCH REC Record n does not exist.

--EOF HIT The end-of-file has been encountered.

--OVERFLOW Commentary typed in has overflowed past column 140 with nonblank characters.

SE Set Intrarecord Mode

SE causes Edit to accept successive lines of intrarecord commands. The SE command has the format shown below.

*SE n[-m][,c[,d]]

Each input line of intrarecord commands is applied, in order, to columns c through d of every record in the range n through m. If m is missing, only record n is processed. The default values for c and d are 1 and 140 respectively.

If several commands are entered on one line, all commands on the line are executed on one record before the next record is processed. The first occurrence of a file or record-editing command terminates the effect of the SE command. All commands executed in the intrarecord mode apply only to the strings lying entirely within columns c through d.

SE may be used on the same input line with other intrarecord commands, but when so used, it must be the first command on the line.

Edit responds to user errors with the following messages:

-BAD COL. NO. PAIR The columns specified are not in range 1 through 140, or $c \leq d$.

-P1:NO SUCH REC Record n does not exist.

--EOF HIT The end-of-file was encountered.

-Cn:COMMND ILGL HERE The nth command of the input line is not an intrarecord command; the intrarecord mode is terminated.

SS Set and Step

SS causes Edit to start at a specified record and proceed to each record in succession, accepting one line of intrarecord commands to update the current record. The SS command has the format shown below.

*SS n[,c[,d]]

The first record to be updated has the sequence number n. Intrarecord commands will only be effective on strings that lie wholly within columns c through d. The default values for c and d are 1 and 140, respectively.

Edit prompts for commands for each successive record with the sequence number, followed by a double asterisk. The SS command is terminated by typing a null record in place of an intrarecord command.

Edit responds to user errors with the following messages:

-BAD COL. NO. PAIR The columns specified are not in range 1 through 140, or $c \leq d$.

-P1:NO SUCH REC Record n does not exist.

--EOF HIT The end-of-file was encountered.

-Cn:COMMND ILGL HERE The nth command of input line is not an intrarecord command; the "set and step" mode is terminated.

ST Set, Step, and Type Record

This command is similar to SS except that the contents of each record is typed, along with its sequence number, prior to accepting a command. The ST command has the format shown below.

*ST n[,c[,d]]

The parameters of the command and the error messages which Edit types are the same as those for the SS command.

INTRARECORD EDITING COMMANDS

The intrarecord commands make changes within an individual record. They generally manipulate character strings. These commands may only be given after the user selects an intrarecord mode with the SE, SS, or ST commands.

The intrarecord commands will be discussed in the following order:

S	Substitute string.
D	Delete string.
F	Insert string following.
P	Insert string preceding.
O	Overwrite string.
E	Overwrite string; blank fill.
R and L	Shift string.
TS, TY	Type individual records.
JU	Jump to new sequence.
NO	No change.
RF	Reverse blank preservation flag.

Commands in the intrarecord group may be linked together on a single line through use of the semicolon (;). The following command sequence would select a line, type the original, edit, and type the new version:

*SE 100; TY; /TEMP/S/B/;/JK/F/+BETA/;TY ⏎

The following conventions are used with intrarecord commands:

1. j/string/x

   means that command x is to operate on the jth occurrence of the indicated string found between columns c through d as specified by an SE, SS, or ST command. If j = 0, this means that the command is to operate on all occurrences of the string between columns c and d. If j is missing, the default is 1. A single / may be included in the string by typing two slashes in succession.

2. k x

   means that command x is to operate on the character contained at column k, where k must lie between columns c and d of the SE, SS, or ST command.

The following general errors are possible:

-MISSING SE No SE command was given. Either an SE, SS, or ST command must be given in response to this message.

--Cn:NO SUCH STRG The string referred to by the nth command of the input line does not exist between columns c and d. When the SE command operates on a range of lines, this message will be given once if the condition occurs at any time during scanning of the range.

--Cn:COL > LIMIT The value specified for k is greater than d for the nth command.

--Cn:COL < LIMIT The value specified for k is less than c for the nth command.

Before reading the intrarecord command descriptions, it is important to note the following information:

Note: In any intrarecord command that seeks a matching string in the image, only those strings that lie totally within the specified column bounds will be found. Partial matches to a column boundary will be ignored. In subsequent examples, references to columns c and d pertain to the column boundaries given in the SE, SS, or ST command.

S String Substitution

S causes Edit to locate a specified string ($string_1$) between columns specified by an SE, SS, or ST command and replace it with another string ($string_2$). The S command has the format shown below.

*j /$string_1$/S/$string_2$/

The image to the right of $string_1$ is adjusted right or left as required, if the lengths of $string_1$ and $string_2$ differ. $String_2$ may extend past column d if d < 140.

If j = 0, all occurrences of $string_1$ between columns c and d are replaced by $string_2$. Otherwise, only the jth occurrence is replaced. If j is missing, the default value is 1.

Example:

Command	Effect		
*/LW/S/CW/		LW,R5 ALPHA+2 CW,R5 ALPHA+2	old new
*/10/S/5/		LW,R10 B LW,R5 B	old new
*/$10/S/ENTRY/	$10 ENTRY	LW,R5 ALPHA LW,R5 ALPHA	old new
*/ALPHA/S/B/		LW,R5 ALPHA+2,R6 LW,R5 B+2,R6	old new
*2/5/S/55/	15	C=DISQRT(TEMP+2.5 *BASE)	old
	15	C=DSQRT(TEMP+2.55 *BASE)	new

If nonblank characters overflow beyond column 140 of the nth command, Edit prints the message:

```
--Cn:OVERFLOW
```

D Delete String

D causes Edit to locate a given occurrence of an indicated string, between columns specified by an SE, SS, or ST command, and delete it. The D command has the format shown below.

```
*[j]/string/D
```

If j = 0, all occurrences of the string between c and d are deleted. Otherwise, only the jth occurrence is deleted. If j is omitted, the default value is 1.

Example:

```
*EDIT SOMEFILE (RET)
*TY 7 (RET)
    7.000    STW,4    ALPHA    ANSWER
*SE 7 (RET)
*/ANSWER/D (RET)
*TY 7 (RET)
    7.000    STW,4    ALPHA
```

P Precede String

P causes Edit to start before the first character of a given occurrence of a specified string ($string_1$) or column k and insert another string ($string_2$), pushing characters of the first string to the right as required to make room. The P command has the format shown below.

```
*[j]/string1/P/string2/
```

or

```
*kP/string2/
```

$String_2$ may legally extend beyond column d if d < 140. The first character of $string_2$ will occupy the column vacated by the first character of $string_1$, etc.

If j = 0, Edit will insert $string_2$ before all occurrences of $string_1$ between columns c and d. However, after $string_1$ has been found once and $string_2$ inserted before it, scanning for the next occurrence resumes at the next character after $string_1$, as adjusted by the insertion. If j is not equal to zero, the command will only affect the jth occurrence of $string_1$. If j is omitted, the default value is 1.

Example:

```
*SE 17.69 (RET)          (set intrarecord mode)
*TS;0/AA/P/./;TS (RET)   (type; edit; type)
AAAAAAA                  (original record)
.AA.AA.AAA               (edited record)
```

If nonblank characters overflow beyond column 140 by the action of the nth command, Edit prints the following message:

```
--Cn:OVERFLOW
```

F Follow By

F causes Edit to start after the last character of a given occurrence of a specified string ($string_1$) or column k and insert another string ($string_2$), pushing everything from this column right as required to make room. The F command has the format shown below.

```
*[j]/string1/F/string2/
```

or

```
*kF/string2/
```

The j specifies that the jth occurrence of $string_1$ between columns c and d (specified by an SE, SS, or ST command) is to be followed by $string_2$. If j is omitted, the default value is 1. In the case where j = 0, the Editor inserts $string_2$ at all occurrences of $string_1$ between columns c and d. Scanning for the next occurrence of $string_1$ resumes following the last character of $string_2$. If a given occurrence of $string_1$ is shifted beyond column d due to previous insertions, it will not be scanned.

$String_2$ may legally extend past column d if d < 140.

Example:

Command	Effect		
*_/AB/F/+2/	LW,R6 LW,R6	AB,R2 AB+2,R2	old new

If nonblank characters overflow beyond column 140 by action of the nth command, Edit prints the following message:

--Cn:OVERFLOW

O Overwrite

O causes Edit to start at the column occupied by the first character of a given occurrence of a specified string ($string_1$) or column k and overwrite with another string ($string_2$). No blank preservations or other adjustment is done and all columns not overwritten remain unchanged. The O command has the form shown below.

*_[j]/$string_1$/O/$string_2$/

or

*_kO/$string_2$/

$String_2$ may overwrite beyond column d if d< 140. The j specifies that the jth occurrence of $string_1$ between affected columns is to be overwritten by $string_2$. If j is omitted, only the first occurrence is overwritten. If j = 0, all occurrences are overwritten. In the case where j = 0, $string_2$ is not scanned by Edit after $string_1$ is overwritten. Edit begins scanning with the column following $string_2$.

If nonblank characters overflow beyond column 140 by action of the nth command, Edit prints the following message:

--Cn:OVERFLOW

E Overwrite and Extend Blanks

E causes Edit to start at the column occupied by the first character of a given occurrence of a specified string ($string_1$) or column k and overwrite with another string ($string_2$). The E command has the format shown below:

*_j /$string_1$/E/$string_2$/

or

*_kE/$string_2$/

Blanks are extended from the end of $string_2$ through column d (where d is the upper limit of the column range selected by an SE, SS, or ST command). $String_2$ may overwrite beyond column d if d < 140, but blank extension only occurs through column d.

The j specifies that the jth occurrence of $string_1$ between affected columns is to be overwritten by $string_2$. If j is omitted, only the first occurrence is overwritten. The specification j = 0 may not be specified, since blank extension precludes multiple substitutions within the same record.

Edit responds to user errors with the following messages:

--Cn:'ALL' IGNORED The specification j = 0 was used, but since it is not meaningful for E, j = 1 was substituted.

--Cn:OVERFLOW $String_2$ overflowed past column 140 with nonblank characters.

R and L Image Shifting

R and L commands cause portions of the record image to be shifted right (R) or left (L). The R and L commands have the form shown below.

*_[j]/string/{R L}s

or

*_k{R L}s

The string must lie wholly within columns c and d specified by the current SE, SS, or ST command. The specified substring may contain embedded blanks, but the string to be shifted terminates with the first blank following the specified substring.

The j specifies that the jth occurrence of the specified substring between affected columns is to be shifted, together with all subsequent contiguous nonblank characters. If j is omitted, only the first such occurrence is shifted. Note that j = 0 may not be specified for this command.

L Shift Left

The jth field that begins with the indicated string (or column k) is shifted left s positions. If blank preservation (see the BP command) is ON, all of the fields to the right of the string are shifted left, intact, and the fields to the left of the string are overwritten (i.e., destroyed). If blank preservation is OFF, blanks are inserted to the right of the jth field, and the fields to the left of the string are overwritten. The shift may legally overwrite below column c.

R Shift Right

The jth field that begins with the indicated string (or column k) is shifted right s positions. If blank preservation is ON, blanks are inserted to the left of the string and all of the fields to the right of the string are shifted right, intact. If blank preservation is OFF, blanks are inserted to the left of the string and are removed to the right. With blank preservation OFF, the image area to the right of the string may be

compressed, but at least one blank will be left between each field; that is, overwriting does not occur in a shift right. The shift may legally push characters beyond column d, if d is less than 140.

In the following examples, blank preservation is OFF.

Command	Effect			
*/L/R1	$10	LW,R6	B	old
	$10	LW,R6	B	new
*/L/R9	$10	LW,R6	B	old
			LW,R6 B	new
*/L/L1	$10	LW,R6	B	old
		LW,R6	B	new

Edit responds to user errors with the following messages:

--Cn:'ALL' IGNORED The value j = 0 was specified but since it is not meaningful for R and L, j = 1 was substituted.

--Cn:UNDERFLOW Characters were lost to the left of the record.

--Cn:OVERFLOW Characters were lost to the right of the record.

TS Type, Suppressing Sequence Number

TS causes Edit to type the contents of the record currently open for editing under control of an SE, SS, or ST command. (Unlike the record-editing version, the intrarecord version of TS does not allow column specification.)

The TS command has the format shown below.

*[. . .;] TS[; . . .]

The three dots indicate that intrarecord commands may precede or follow the TS command.

Example:

```
*SE 5; TS (RET)
L1  LW,5 K
*19O/KLB/; TS; 37O/GET KLB/; TS (RET)
        (overwrite, type, overwrite, type)
L1  LW,5  KLB
L1  LW,5  KLB      GET KLB
```

Because all commands on a single input line are executed for the first record before the second record is processed, etc., TS will type each line in turn after all editing up to the TS command has been done.

Example:

```
*SE    10- 10.2 (RET)
*2/A/F/,4/;TS (RET)
    DATA,4    X'FF'          (10.0)
    DATA,4    0.5            (10.1)
    DATA,4    GQX,X'0B'      (10.2)
```

TY Type, Including Sequence Number

TY is the same as TS, except that each line is printed with its sequence number. (Unlike the record-editing version, the intrarecord version of TY does not allow column specification.)

The TY command has the format shown below.

*[. . .;] TY[; . . .]

JU Jump

JU causes the SS or ST command to jump to a specified record and then continue stepping from that point. JU may only be used while in the "step" mode (i.e., while under the control of an SS or ST command). The JU command has the form shown below.

*[. . .;] JU n

Record n may be forward or backward from the current sequence number at the time JU is given. The dots indicate that JU may be used on compound lines (i.e., a line with more than one command on it), but in such a case JU must be the last command on the line.

If the record specified by n does not exist, Edit prints the following message:

--Cn: NO SUCH REC

NO No Change

NO may be used only while in the "step" mode and specifies that no editing is desired on the current active line under the set. The NO command has the format shown below.

*NO

Example:

```
*ST 27.5 (RET)
27.500                  LW,6    BLK
*NO (RET)
30.000                  STW,6   ALT
*/ALT/F/+ 19 ; TY ; JU 34 (RET)
30.000                  STW,6   ALT + 19
34.000                  AI,F    X'91'
*
```

RF Reverse Blank Preservation Mode

RF causes the current setting of the blank preservation mode (see the BP command) to be reversed temporarily. The RF command has the form shown below.

*[. . .;] RF ; . . .

or

*. . . ; RF[; . . .]

The mode is reversed only for the duration of the input line in which RF appears and only for those commands which follow the RF command, and blank preservation is restored to its initial setting when a new input line is entered (i.e., at the time a new prompt character is given). Thus, to have any effect, RF must always be used as part of a compound input line and must be followed by other commands.

Example:

```
*SE    10; TY (RET)
10.000 L5     LW,4 X GET CURRENMT ADDR
*RF;/NM/S/N/;TY (RET)
10.000 L5     LW,4 X GET CURRENT ADDR
```

Without using RF in this case (assuming that BP OFF is the initial setting), one would get two blanks after CURRENT. In all cases, the BP mode is restored to the value it had before any RF commands were given.

MESSAGES

During the course of executing any command, Edit may communicate with the user through a variety of messages.

Possible messages are summarized in Table 21. The following conventions are used in regard to message formats:

1. A message preceded by two periods is a comment on some system-oriented operation. For example,

   ..COPY DONE

2. A message preceded by two minus signs indicates the occurrence of some event (during the execution of a command) of which the user should be aware; the command is not aborted. For example,

   --EOF HIT

3. A message preceded by a single minus sign is an error message describing a condition that aborts the current command and causes any others on the same line to be skipped. For example,

   -P1:NO SUCH REC

Such a message is particularized as to cause by the following prefixes:

Prefix	Cause of Error
-Ck:	The kth command of the previous line caused the error.
-Pk:	The kth parameter of the first command on the previous line caused the error.
-CkPj:	The jth parameter of the kth command of the previous line caused the error.

EDIT COMMAND SUMMARY

Table 22 is a summary of Edit commands. The left-hand column gives the command formats. The right-hand column gives the command function and options.

Table 21. Edit Messages

Message	Meaning
--xxx RECS DLTED	The indicated number of records have been deleted.
-BAD COL. NO. PAIR	The columns specified are not in the range 1 through 140, or c > d.
--Cn: 'ALL' IGNORED	The value 0 was specified for j. Since this value is not meaningful for the command, the value 1 has been assumed.
-Cn: COMND ILGL HERE	The nth command of the previous line is invalid and the intrarecord mode has been terminated.
--Cn: NO SUCH STRING	A specified string was not found and no substitution was made in processing the nth command of the previous line. Processing continues. When the SE command operates on a range of lines, this message will be given once if the condition occurs at any time during scanning of the range.
--Cn: OVERFLOW	The nth command of the previous line has caused characters to be shifted past column 140. Processing continues.

Table 21. Edit Messages (cont.)

Message	Meaning
--Cn: UNDERFLOW	Characters were lost to the left of the record.
..COPY DONE	A COPY operation has been completed.
..COPYING	A COPY operation has begun.
--CUTOFF AT x(y)	A specified operation could not be completed because of a conflict between an existing sequence number and a new one. The value x is the current sequence number of the last record affected (formerly record y).
..DELETED	A specified file has been deleted.
--DONE AT x	A specified operation has been completed. The value x is the current sequence number of the last record affected.
..EDIT STOPPED	The record editing mode has been terminated.
--EOF HIT	One or both sequence numbers specified are higher than the highest one in the file.
-FILE EXISTS: CAN'T BUILD	An existing file has the same name as that specified in a BUILD command.
-FILE NOT KEYED:MUST COPY	A specified file has no sequence numbers. The file must be copied with sequencing specified (via the COPY command).
-MERGE DESTINATION NOT KEYED	The destination file in a MERGE command is not keyed. The file must be copied with sequencing specified.
-MERGE SOURCE NOT KEYED	The source file in a MERGE command is not keyed. The file must be copied with sequencing specified.
..MERGE STARTED	A MERGE operation has begun.
-MISSING SE	No SE, SS, or ST command is currently in effect. The specified intrarecord task has been aborted.
-NO SUCH FILE	A specified file does not exist.
--NONE	There were no records in the specified range containing the indicated string.
-NOT F/M/S	A parameter other than F, M, or S has been specified in a TA command.
-NOT ON/OFF	A parameter other than ON or OFF has been specified in a BP or CR command.
--NOTHING TO DE	No records (to be deleted) were found in the specified range.
--NOTHING TO MOVE	No records (to be moved) were found in the specified range.
--OVERFLOW	More than 140 characters have been typed on a line or characters have been shifted past column 1 or 140. Excess characters are lost.
-P1: FILE NOT KEYED & P3 NULL	A file to be copied has no sequence numbers and no sequencing has been specified. The COPY operation has been aborted.
-P1: NO SUCH FILE	A COPY command has specified that a nonexistent file is to be copied.
-P1: NO SUCH REC	A specified record does not exist. The command has been aborted.
-P2: COL ERROR	Column c is greater than 140.

Table 21. Edit Messages (cont.)

Message	Meaning
-P2: FILE EXISTS	A COPY ON command specified the name of an existing file.
-P2: REC EXISTS	A specified record already exists. The command has been aborted.
--RNG OVERLAP	Specified ranges of sequence numbers overlap. The command has been ignored.

Table 22. Edit Command Summary

Command	Description
BP [{ON / OFF}]	Sets the blank preservation mode. When "on", all strings of blanks are preserved during intrarecord operations. When "off", blank strings are compressed to a single blank or expanded as required to retain column alignment of nonblank fields. The default mode is "off".
BUILD fid[,n[,i]]	Enables the user to create a new file. Options: n is the sequence number at which the new file is to start. The default value is 1. i is the value by which the sequence numbers are to be incremented. The default value is 1.
CM n,c	Causes Edit to insert commentary (given by the user) into specified columns (starting at column number c) of each successive record beginning at the specified sequence number n.
COPY fid_1 {ON / OVER} fid_2[,n[,i]]	Copies a file. Fid_2 identifies the file to which fid_1 is to be copied. Options: n is the starting sequence number for the new file. If omitted, the sequence numbers of fid_1 are retained in the copy. i is the sequence number increment for the new file. The default value is 1.
CR {ON / OFF}	Controls the inclusion of the carriage return character (X'15') at the end of each record in the user's output file. ON includes the X'15' terminator in the output file. OFF excludes the X'15' terminator from the output file and is the default setting.
[j]/string/D	Locates a given occurrence of the indicated string, between columns specified by an SE, SS, or ST command, and deletes it. Options: j specifies that only a particular occurrence (the jth occurrence) of the string in the specified columns is to be deleted. If j equals zero, all occurrences of the string in the specified columns are to be deleted. If j is omitted, the default value is 1.
DE n[-m]	Deletes all records whose sequence numbers lie in a specified range beginning at n. Option: m indicates the number of the last record to be deleted. If m is omitted, only record n will be deleted.
DELETE fid	Deletes the file specified by fid from the log-on account.

Table 22. Edit Command Summary (cont.)

Command	Description
[j]/string$_1$/E/string$_2$/ or kE/string$_2$/	Starts at a column occupied by the first character of a given occurrence of a specified string (string$_1$) or column k and overwrites with another string (string$_2$). Blanks are extended from the end of string$_2$ through column d (which is specified in an SE, SS, or ST command.) Option: j specifies that the jth occurrence of string$_1$ between affected columns is to be overwritten by string$_2$. If j is omitted, only the first occurrence is overwritten; j may not be zero.
EDIT fid	Opens a file to be edited and enters the record editing mode.
END	Closes all active files and returns control to the terminal executive language (TEL).
[j]/string$_1$/F/string$_2$/ or kF/string$_2$/	Starts after the last character of a given occurrence of a specified string (string$_1$) or column k and inserts another string (string$_2$), pushing everything from this column right as required to make room. Option: j specifies that the jth occurrence of string$_1$ between columns c and d (specified by an SE, SS, or ST command) is to be followed by string$_2$. If j is omitted, the default value is 1. If j equals zero, string$_2$ is inserted at all occurrences of string$_1$ between columns c and d.
FD n[-m],/string/[,c[,d]]	Searches for the specified string between specified columns in a specified range of records beginning at record n. If the string is found, the record containing it is deleted from the file. Options: m specifies the sequence number of the last record to be searched. If omitted, only record n is searched. c specifies the lowest column number of the field to be searched. The default value is 1. d specifies the highest column number of the field to be searched. The default value is 140.
FS n[-m],/string/[,c[,d]]	Searches for the specified string between specified columns in a specified range of records beginning at record n. Each time the string is found, the sequence number of the record is printed. Options: Same as for FD.
FT n[-m],/string/[,c[,d]]	Searches for the specified string between specified columns in a specified range of records beginning at record n. Each time the string is found, the sequence number and the contents of the record are printed. Options: Same as for FD.
IN n[,i]	Inserts new records into a file starting at record n. Edit prompts the user with the sequence number of each record to be inserted. Option: i specifies an increment amount for successive record numbers. If i is omitted, the increment size specified in the most recent record editing command is used. If no such command has been given, the default value is 1.

Table 22. Edit Command Summary (cont.)

Command	Description
IS n[,i]	Inserts new records into a file starting at record n. Edit does not prompt with sequence numbers of the records to be inserted. Option: i specifies an increment amount for successive record numbers. If i is omitted, the increment size specified in the most recent record editing command is used. If no such commands have been given, the default value is 1.
[. . . ,] JU n	Causes the SS or ST command to jump to the specified record n and then continues stepping from that point. Option: The dots indicate that JU may be used on a line with more than one command on it, but in such a case JU must be the last command on the line.
MD n[-m],k[-p][,i]	Moves records within a file from a range beginning at n to a range beginning at k. Records originally in the destination range are deleted. Options: m specifies the sequence number of the last record that is to be moved. If omitted, only record n is moved. p specifies the upper limit of the range of records to be deleted. If omitted, only record k is deleted. However, records from the range n-m are still moved to record k and following. i specifies the increment value to be used for renumbering records. If omitted, the most recent value specified in a record edit command is used. If no such commands have been given, the default value is 1.
MERGE fid_1 [,n_1[-n_2]]INTO fid_2, n_3[-n_4][,i]	Merges records from fid_1 into fid_2. The records are numbered beginning at n_3 in fid_2. Options: n_1 specifies the number of the first record in fid_1 to be merged. If omitted, all records of fid_1 are to be merged. n_2 specifies the number of the last record in fid_1 to be merged. If omitted, only record n_1 is merged. n_4 specifies the number of the last record in fid_2 which is to be replaced by merged records. If omitted, only record n_3 is replaced by a merged record. i specifies an increment amount for resequencing from n_3. The default value is 1.
MK n[-m],k[-p][,i]	MK is identical to MD except that the records in the range n-m are not deleted as they are moved. Options: Same as for MD.
NO	Specifies that no editing is to be performed on the current active line.
[j]/$string_1$/O/$string_2$/ or kO/$string_2$/	Starts at the column occupied by the first character of a given occurrence of a specified string ($string_1$) or column (k) and overwrites with another string ($string_2$). Option: j specifies that only a particular occurrence (the jth occurrence) of the string is to be overwritten. If j equals zero, all occurrences of the string are to be overwritten. If j is omitted, the default value is 1.

Table 22. Edit Command Summary (cont.)

Command	Description
[j]/string$_1$/P/string$_2$/ or kP/string$_2$/	Starts before the first character of a given occurrence of a specified string (string$_1$) or column k and inserts another string, pushing characters of the first string to the right as required to make room. Option: Same as for the O command.
[j]/string/{R / L}s or k{R / L}s	Shifts portions of the record right (R) or left (L) the number of positions indicated by s. The field to be shifted begins with the indicated string. Option: j specifies that the jth occurrence of the specified substring between affected columns is to be shifted, together with all subsequent contiguous nonblank characters. If j is omitted, only the first such occurrence is shifted. Note that j = 0 may not be specified for this command.
[. . . ;] RF ; . . . or . . . ; RF[; . . .]	Causes the current setting of the blank preservation mode ("on" or "off") to be reversed temporarily (for the current line only). Option: The dots indicate that other commands are present on the line.
RN n,k	Renumbers a specified record from number n to number k.
[j]/string$_1$/S/string$_2$/	Locates a specified string (string$_1$) between columns specified by an SE, SS, or ST command and replaces it with another string (string$_2$). Option: j specifies that only a particular occurrence (the jth occurrence) of string$_1$ is to be replaced. If j equals zero, all occurrences of string$_1$ are to be replaced. If j is omitted, the default value is 1.
SE n[-m][,c[,d]]	Causes Edit to accept successive lines of intrarecord commands to be applied to records beginning at record n. Options: m specifies the number of the last record to which the intrarecord commands are to be applied. If omitted, the intrarecord commands are only applied to record n. c specifies the smallest column number of the range of columns to which the intrarecord commands are to be applied. The default value is 1. d specifies the largest column number of the range of columns to which the intrarecord commands are to be applied. The default value is 140.
SS n[,c[,d]]	Causes Edit to start at a specified record (record n) and proceed to each record in succession, accepting one line of intrarecord commands to update the current record. Options: c specifies the smallest column number of the range of columns to which the intrarecord commands are to be applied. The default value is 1. d specifies the largest column number of the range of columns to which the intrarecord commands are to be applied. The default value is 140.
ST n[,c[,d]]	Causes Edit to start at a specified record (record n) and proceed to each record in succession, accepting one line of intrarecord commands to update the current record.

Table 22. Edit Command Summary (cont.)

Command	Description
ST n[,c[,d]] (cont.)	The sequence number and contents of each record is typed prior to accepting a command. Options: Same as for the SS command.
TA{F / M / S}	Causes Edit to set or reset the terminal tab stops. F implies FORTRAN and a tab set at column 7. M implies Meta-Symbol and tabs set at columns 10, 19, and 37. S implies Meta-Symbol, short form, and tabs set at columns 8, 16, and 30.
TC n[-m][,c[,d]]	Types the sequence numbers and the contents of specified columns of one or more records beginning at record n. Any nonblank strings within the columns typed are shifted to the left to compress each blank string to a single blank. Options: m specifies the number of the last record to be typed. If omitted, only record n is typed. c specifies the smallest column number of the range of columns to be typed. The default value is 1. d specifies the largest column number of the range of columns to be typed. The default value is 140.
[. . . ;] TS[; . . .]	Types the contents of the record currently open for editing under control of an SE, SS, or ST command. Option: The ellipses (dots) indicate that other commands may be present on the line.
TS n[-m][,c[,d]]	Types the contents of specified columns of one or more records beginning at record n. Options: Same as for the TC command.
[. . . ;] TY[; . . .]	Types the sequence number and contents of the record currently open for editing under control of an SE, SS, or ST command. Option: The ellipses (dots) indicate that other commands may be present on the line.
TY n[-m][,c[,d]]	Types the sequence numbers and the contents of specified columns of one or more records beginning at record n. Options: Same as for the TC command.

7. DELTA

INTRODUCTION

Delta is designed to aid in the on-line debugging of programs at the assembly-language or machine-language levels. It operates on object programs and tables of internal and global symbols used by the program but does not require that the tables be at hand. With or without the symbol tables, Delta recognizes computer instruction mnemonic codes and can assemble machine-language programs on an instruction-by-instruction basis. The main purpose of Delta, however, is to facilitate the activities of debugging by

1. Examining, inserting, and modifying such program elements as instructions, numeric values, and coded information (i.e., data in all its representations and formats).

2. Controlling execution, including the insertion of breakpoints into a program and requests for breaks on changes in elements of data.

3. Tracing execution by displaying information at designated points in a program.

4. Searching programs and data for specific elements and subelements.

Although Delta is specifically tailored to machine language programs, it may be used to debug programs written in FORTRAN, COBOL, or any other language. Delta is designed and interfaced to UTS in such a way that it may be called in to aid debugging at any time, even after a program has been loaded and execution has begun.

The command language of Delta is cryptic and highly encoded, but is easily learned and used by the professional programmer. It is similar to the DDT (Dynamic Digital Debugging Tool) language family that has been used on a variety of machines for the last decade.

There are two versions of Delta:

1. A user version with codes and restrictions appropriate to multiple on-line users operating in the slave mode from on-line terminals.

2. An executive version for system debugging that operates in executive mode under control of one of the operator's consoles.

Differences in the language syntax of the two versions are few and are noted in this chapter. The main orientation of the chapter, however, is towards the user version of Delta. Instructions for calling the executive version of Delta are given in the UTS/Preliminary Technical Manual.

CALLING DELTA

The user version of Delta may be brought in at the time the user loads his program into core for execution or by direct call after execution begins. Delta also may be brought in without prior program loading for writing and checking short Meta-Symbol or machine language programs. The commands used to call Delta are at the executive level:

1. To bring in Delta at program load time, the user gives the command "RUN rom UNDER DELTA" or the command "START lmn UNDER DELTA". Control goes to Delta, and the user may examine and modify his program before passing control to it.

2. To bring in Delta after a program has started, the user returns to the executive level by using the terminal command Y^C (by simultaneously pressing the control shift and the Y keys) and then giving the executive command DELTA.

   Note: Attempting this approach may result in the message:

   A100 DON'T TRY TO DEBUG A SHARED PROCESSOR

   This means that execution of the user's program had not actually begun when the Y^C command was given and a processor such as LINK was operational instead. If this happens, the user may either re-try this approach and wait for execution of this program to actually begin or use the approach outlined in 1 (bringing Delta in at program load time).

3. To bring in Delta without prior program loading, the user simply gives the executive command DELTA. Writing programs with Delta is discussed at the end of this chapter.

Delta responds to these commands by typing "DELTA HERE". In the user version, it follows this by its prompt character, the bell. (The executive version of Delta does not have a prompt character, nor does the user version when connected to a keyboard display.) Delta is then ready to accept a command.

EXITING DELTA

There is no Delta command that concludes Delta and returns control to TEL. Therefore, the Y^C control combination must be used to return control to TEL.

PREREQUISITES

There are three Delta restrictions the user must be aware of when he writes a program that will be run under Delta:

1. All assembly language mnemonics are reserved words in Delta and may not be used as symbols for instruction or data tags.

2. The symbols used for the program must follow the rules for symbols in Meta-Symbol, except that the first seven characters of all the symbols must be unique. This is necessary because symbols are carried in Delta's symbol table as seven characters. Symbols with more than seven characters are truncated to include only the first seven. Thus, symbols that originally were longer than seven characters are indistinguishable from each other if the first seven are the same. (If this happens, only the last definition is retained.)

3. A Meta-Symbol program should be assembled with the option SD if the user wishes to use internal symbolic references while debugging. (The SD option causes the assembler to produce debugging object code for internal symbols for use with Delta.) Also, at run time the user must request that the internal symbol table with this code be made available to Delta by using the Delta command s;S where s is the name of the file. (See the section "Symbol Table Control".)

SAVING PROGRAM MODIFICATIONS

When a user debugs a program under Delta, he may make modifications to the program code and symbol tables. These modifications only affect the core image of the program and are not saved unless the user returns control to TEL (by issuing the Y^C command or by depressing the break key four times) and then issues a SAVE command. (See SAVE command in Chapter 3.) A program that contains overlays cannot be saved as one intact program and therefore this approach to making the modifications permanent is not applicable.

CONVENTIONS

The following conventions are used in explaining the format of the commands typed by the user:

1. Special characters, numbers, and uppercase letters stand for themselves. Thus, in the command e;G the user actually types the semicolon and the G.

2. Lowercase letters are used to indicate places where the user has a choice of things to type. The letter e, alone or postscripted, is used to represent any expression consisting of symbols, special symbols, instruction mnemonics constants, the operators plus (+) and minus (-) and space ( ). At times, other lowercase letters are used to stand for expressions when some additional mnemonic content seems desirable (e.g., n, loc, val, m).

3. The letter f stands for one of the format characters. (The format codes are listed in Table 23.)

4. Abbreviations for user keystrokes are as follows:

Characters Used in Text	User Delta Keystroke	Executive Delta Keystroke
RET	RETURN	RETURN
LF	LINE FEED	EOM
↑	SHIFT and N	&
\	SHIFT and L	¢
TAB	CTRL and I	TAB
BRK	BREAK	Sigma INTERRUPT Switch

Table 23. Format Codes

Code	Meaning
F	Symbol table specified format.
X	Hexadecimal word.
I	Signed decimal integer
C	EBCDIC characters.
R	Symbolic instructions with symbolic addresses.
A	Symbolic instructions with hexadecimal addresses.
S	Short floating-point number.†
L	Long floating-point number.†

†User version only; both have the format XXXXX E ±YY

COMMAND DELIMITERS

The characters listed below are used as end-of-message characters and, in most cases, as commands in the Delta language. Each has a particular meaning that will be discussed in detail with the commands to which it applies.

/

=

RET

LF (This character is represented by the EOM key in the executive version of Delta.)

↑ (This character is represented by the & key in executive Delta.)

TAB

With the exception of the slash (/) and equal (=) characters, which interact immediately within a single typed line, these characters cause a carriage return and a line feed.

More than one command can be input on a command line by separating the commands with spaces. The following line contains five commands:

PROG;S TAG1;B TAG2;B ;B ;D (RET)

However, it is important to note that any command that changes the contents of a cell should be the last command on a multiple-command line.

CORRECTING TYPING ERRORS

Correcting typing errors while using Delta requires special consideration because the = and / characters cause immediate interaction with UTS without proceeding to a new line. The RUBOUT key will not affect a / or = character nor any information which precedes it on a line. Canceling a line by simultaneously pressing the ESC or CONTROL key and the X key may only cancel a partial line. If a / or = character appears in the line, that character and all characters preceding it will not be canceled.

In the executive version of Delta, the question mark (?) cancels the command line and the at sign (@) is the rubout command.

EXPRESSIONS

Expressions are typed by the user for location value, for parameter value, and for assembly into an instruction. Expressions are composed of

Program symbols.

Special symbols (see Table 24).

Assembly language mnemonics.

Explicit constants.

The operators plus (+) and minus (-).

Space ( ).

Examples:

```
;I
.1E07
A
A+3
A+3-B
AI,1 2
STW,7    *LOC
LW,7    TAB,5
CAL1,3    LIST
```

Table 24. Special Symbols

Symbol	Meaning
$ or .	Last opened cell address.
;I	Instruction counter } As set by the last entry
;C	Condition code } to Delta or as changed
;F	Floating controls } by the user.
;M	Search mask.
;1	Lower search bound.
;2	Upper search bound.
;Q	Last quantity typed.

The space character is used to introduce the address field in expressions to be assembled into instructions.

CONSTANTS

Constants must be input in the following formats:

1. Hexadecimal - hexadecimal numbers preceded by a period.

   .1C28

   .BE3

   .FFFFFFFF

2. EBCDIC - EBCDIC characters surrounded by single quotes. (EBCDIC text strings must consist of no more than four characters. If fewer than four characters are specified, the characters are right-justified and zero-filled.)

   '%ERR'

   'LOC2'

   'TOM'

3. Decimal - numerics only.

   1234

   -250

   10

Hexadecimal and decimal constants are output in the same format that they are input. EBCDIC constants are output as EBCDIC characters without the surrounding single quotes. Non-printing EBCDIC characters may also be output, including the EOT (end of transmission (04)) character, which will turn off some types of terminals.

DELTA COMMANDS

EXPRESSION EVALUATION: THE = COMMAND

Expressions consisting of program symbols, special symbols, assembly language mnemonics, explicit constants, and the operators plus (+) and minus (-), and space ( ) may be evaluated by use of the = command. When a program symbol is evaluated, the result is its absolute address. When a special symbol is evaluated, the result is the value that the symbol is set to. When an explicit constant is evaluated, the result is its numeric equivalent. When an assembly language instruction is evaluated, the result is its machine language equivalent.

The basic = command is

e =

Note that no carriage return is given. Delta responds immediately to the = character and evaluates the expression e.

In this command, no format is specified for typing the expression evaluation, so the default format is used. Usually the default format is X (hexadecimal), but the default can be changed by one of the variations of the = command (which will be discussed shortly).

Examples:

```
2+2 = .4
5+5 = .A
TOT = .C12E
AI,6  1 = .20600001
```

The user may temporarily change the output format with the following = command:

```
e(f =
```

where f specifies a particular format code selected from those listed in Table 23, "Format Codes". The temporary change only affects the = command in which it is given.

The default format for output can be changed by the command:

```
(f; =
```

where f specifies a particular format code selected from those listed in Table 23, "Format Codes". The new default will be retained until another (f;= command is given. The original default setting of the output conversion format is X.

Example:

```
5+5 = .A (RET)
(I; =    6+7 = 13   (The default format is changed
                     to integer.)
```

The last expression typed by Delta may be evaluated simply be typing the = character. In the example below, Delta types the expression BAL,5 SUB as a result of the command ALPHA/ (which is discussed in the next section). Then the entire expression BAL,5 SUB is evaluated and the results are typed as a result of the = command.

```
ALPHA/ BAL,5 SUB = .6A5006B3
```

MEMORY CELL OPENING AND DISPLAY: THE /, TAB, and \ COMMANDS

The slash (/) character is a command to Delta to open a memory cell and display its contents. There are several variations of the slash command and these will be discussed below. In each command the cell to be opened and displayed is indicated by an expression (designated by an e in the instruction formats)

The basic / command is

```
e/
```

Note that no carriage return is given. Delta responds immediately to the / character and opens the cell, displaying its contents on the same line. In this command, no format is used. Usually this default format is F (symbol table specified format), but the default can be changed by one of the variations of the slash command (which will be discussed shortly). If the default is F, then the symbol table is searched to find a symbol at the same or at the closest smaller location (within a hexadecimal offset that may be specified by the Delta ;R command discussed later) than the indicated address and the data type associated with the symbol found is used to control output. If no symbol is found within the range, the default is R (symbolic instruction).

Examples:

```
.C125/     .34
A1/        BAL,6  ALPHA
A+1/       STW,5  BETA
BETA/      ABCD
```

The user may temporarily change the output format with the following slash command:

```
e(f /
```

where f specifies a particular format code selected from those listed in Table 23, "Format Codes". The temporary conversion type is retained for all slash commands until the next RET command is given or another format change is specified. (The temporary conversion type is retained over any following LF, ↑, /, and TAB commands.)

Examples:

```
X(X/     .C1     (hexadecimal conversion)
X(C/     A       (EBCDIC character conversion)
X(I/     193     (decimal integer conversion)
```

Below is an example which shows that temporary conversion types are retained until the next RET command is given:

```
D(X/     .3230C122
V/       .2030C122 (RET)     (D,V, and Z are locations
Z/       AI,3   AA (RET)      of program instructions)
```

The default format for output can be changed by the command

```
(f;/
```

where f specifies a particular format code selected from those listed in Table 23, "Format Codes". The new default will be retained until another (f;/ command is given. The original default setting of the output conversion format is F.

Example:

```
X/      .C1 (RET)
(C;/    X/   A
```

The cell addressed by the last expression typed by Delta may be opened and displayed by typing a TAB (produced by simultaneously pressing the CTRL and the I keys for the user version or by pressing the TAB key for executive Delta).

In the example below, the cell DCT8 is opened and displayed.

```
ALPHA/    LW,5    DCT8 (TAB)
DCT8/  .32        (The carriage return was automatically
                   provided by Delta.)
```

The format for display is by default only and is the same default as for the slash command.

If the user types a slash by itself, the cell addressed by the last expression typed by Delta is displayed but not opened. In the example below, ALPHA remains the open cell even though the contents of cell DCT8 are displayed.

```
ALPHA/    LW,5    DCT8    /    .32
```

Conversely, a cell may be opened without displaying its contents by the use of the \ command (produced by simultaneously pressing the SHIFT and the L keys for the user version or by using the £ key for executive Delta). The format of the command is

```
e\
```

In the example below, the cell SUM is opened but not displayed. Then SUM is set to zero.

```
SUM\    0 (RET)
```

Opening a cell without displaying its contents is convenient when the user wishes to insert new contents in memory and is not interested in the current contents.

If the user wishes to store data into a page that is not assigned to his program, the \ command will request that the Monitor assign the page. (This is particularly useful when using Delta to write new programs.) For example:

```
.18000\
```

If the page at .18000 is not assigned to the program, the Monitor will assign it (if possible). Also, the cell at .18000 will be opened.

More than one cell may be displayed by using the following command:

```
e1,e2/
```

where

e1 is an expression which identifies the lower address of a range of cells to be displayed.

e2 is an expression which identifies the upper address of a range of cells to be displayed.

Following the display of these cells, the upper limit cell is open for change. In the following example, ALPHA + 2 is open for change.

```
ALPHA,ALPHA+2/    BAL,4  SUB
ALPHA+.1/         STW,5  DCT2
ALPHA+.2/         AI,6   .100
```

Note that Delta types the word increments as hexadecimal numbers.

Temporary change in the output format may be added to the above command as shown below:

```
e1,e2 (f/
```

where f specifies a particular format code selected from those listed in Table 23, "Format Codes".

Example:

```
100, 101(X/    .58000100
101/           .68000200
```

If the user wishes to interrupt a display that is too long, he presses the BREAK key and the remaining output is discarded. The last displayed cell is opened. The INTERRUPT switch on the Sigma Processor Control Panel accomplishes this in the executive version of Delta.

MEMORY MODIFICATION: THE RET, LF, ↑, AND TAB COMMANDS

These four commands allow the user to store a typed expression for word value into the currently open memory location -- opened by /, \, or one of the modification commands LF, ↑, or TAB. If no expression precedes the command character, the action taken is as described, except that the open cell remains unchanged.

The RET command causes an expression to be assembled and stored in the open memory cell. Carriage return (RET) and new line (LF) characters are sent to the terminal, and temporary display modes are reset to default values. The format of the RET command is

```
e (RET)
```

Example:

```
A/    BAL,4  JWS  BAL,4  GEB (RET)
A/    BAL,4  GEB (RET)
```

(The expression BAL,4 GEB is assembled and stored into A).

```
JED/    EXU  LS  (X/  .68000643/  .78C (RET)
```

(The contents of JED are typed. Then the contents of LS are typed in hexadecimal. JED remains the open cell. Then the contents of .0643 are typed.)

```
./    EXU  LS
```

(The contents of JED are typed. The . is a special symbol (see Table 24) that specifies the last opened cell address.)

Note that a temporary display format was established by the (X/ which carried over until reset by the carriage return (RET) command.

When the user terminates an expression with the new line (LF) command, the value of the expression is stored in the currently open cell, that cell is closed, a new line is produced at the terminal, and the cell with the next higher location value is opened. The type of command used for initial cell opening is preserved and carried forward on succeeding openings as is the display format.

The format of the new line (LF) command is:

e (LF)

Example:

A(I/ 435 436 (LF)
A+.1/ 763 (LF)
(A+.1 is displayed but remains unchanged.)
(A+.2/ 7689 7000 (RET)
EM\ STM,4 ERS (LF)
EM+.1\ BAL,6 LP (LF)
EM+.2\ BGE GAP (RET)

For the executive version the EOM (end-of-message) key replaces the LF key.

The e↑ command is the same as the LF command except that the cell with the next lower location value is opened. For the executive version, & replaces ↑.

Example:

EM+4/ 0 B GW↑
EM+.3/ 0 AI,3 1 (RET)

The TAB command causes the typed expression to be stored in the currently open cell, and that cell is closed. Following output of a carriage return, the cell addressed by the most recently closed cell is opened and displayed (only the address is displayed in the \ mode). The effect is like that of a RET command followed by a ;Q/ command (see Table 24, Special Symbols). The format of the TAB command is:

e (TAB)

The TAB command is useful for patches:

A/ BAL,5 SUB (LF)
A+.1/ STW,6 BETA B PATCH (TAB)
(The carriage return is performed as part of the TAB command.)
PATCH/ .0 AI,6 1 (LF)
PATCH+.1/ .0 STW,6 BETA (LF)
PATCH+.2/ .0 B A+2 (RET)

SYMBOL TABLE CONTROL: THE ;U, ;K, ;S, !, AND <> COMMANDS

There are two types of symbol tables in Delta:

1. Constant (Internal to Delta).
2. User associated or defined.

The first type of table is always present in Delta and consists of the Meta-Symbol instruction mnemonics and a list of special symbols (see Table 24) associated with program debugging.

The second type of table consists of a set of global symbols (those defined by DEF directives) and a set of internal symbol tables, one for each ROM loaded (although some may be combined by Link). (See Chapter 8 on Link.) The internal symbol tables are filed under the name of the file from which the ROM was loaded.

The user must specify that the internal symbol table is to be loaded if he wishes to debug using internal symbolic tags. The command for specifying this is

s;S

where s is the name of the file from which the ROM was loaded. This command causes Delta to load the internal symbols from the program loaded from file s, and these internal symbols replace, for reference purposes, any previously selected internal symbol set. An example of the s;S command is

C=
?2
(Delta is confused because no internal symbol table had been previously loaded. The internal symbol C is not recognized).

BIN;S (RET)
(The internal symbol table is loaded).

C=.C125
(The symbol C is now recognized).

The ;S command alone loads the global symbol table. The user must specify this command if he wishes to debug using global symbolic references.

The user may wish to release to the system the pages used for symbolic tables. The command ;K releases the pages containing the global and internal symbol tables. The command ;KG releases only pages containing the global symbol table and the command ;KI releases only pages containing the internal symbol table. Other uses of the ;K command are

s;K prevents use of the symbol s in constructing output. The symbol is still recognized when typed in. Symbol s is returned to use if the user reloads the symbol table.

;K removes all symbols from the symbol table. The lists of instruction mnemonics and special symbols are not erased. Individual internal symbol tables are recoverable using the s;S command. Global symbols are restored by ;S.

Undefined symbols in the loaded programs are printed by Delta when the ;U command is given. Undefined symbols within the range of an assembler LOCAL directive are lost. They are given a value of zero in the loaded code and do not appear when the ;U command is given.

Symbols may be defined by the user at any time during his debugging session. Symbols so defined are added to the global symbols associated with the program load. Commands for adding symbols to the global symbol table are

s(f! adds the symbol s to the global symbol table with the location value of the currently open cell and format type f. (See Table 23, "Format Codes".) If format type f is omitted, symbolic instruction (R) type is assumed. For example,

C+1/ 0 LOC! LW,4 TAB (RET)

In this example, location C+1 is given the name LOC. Then the contents of LOC are changed to contain the assembled instruction LW,4 TAB.

Note that if a format code is specified for a slash command (e(f/), it is retained until the next carriage return and meanwhile the format specified is applied to any command.

Example:

INST/ CI,2 .4 (RET)
INST(X/ .21200004 SYM! (RET)
INST/ CI,2 .4 SYM/ .21200004

The cell INST was displayed in its default format. The instruction at INST was then displayed in hexadecimal format and then the cell INST was assigned an additional name, SYM, with the display format for SYM being X.

e(f<s>[K] adds the symbol s to the global symbol table with value defined by the expression e and format code f. In addition to the format codes of Table 22, the letter K may be used to indicate value is to be a constant. If f is omitted, R is assumed. If the final angle bracket is followed by a K, the symbol is flagged as a control section type symbol in the symbol table. K may not be used as the format code if K is specified following the final angle bracket.

SINGLE LINE MACROS

Since the symbol table definition gives a 32-bit value to constant symbols, it may be used as a macro-definition facility for single-word values.

Example:

LI,3 0(K<CLEAR>(RET)	The symbol CLEAR now represents the instruction LI,3 0.
AA\ CLEAR (RET)	The cell AA is opened and its content is set to the instruction LI,3 0.
MM\ CLEAR (RET)	The content of MM is also set to the instruction LI,3 0.

EXECUTION CONTROL: THE ;G, ;P, ;X, AND ;) COMMANDS

The four commands described in this section allow the user to begin execution of his program and to resume execution of the program if it is interrupted.

Execution is started by typing the [e];G command,where e is an expression which identifies the starting location. The expression e may be omitted, in which case execution will begin at the first instruction of the program.

Example:

BEGIN;G (RET)
;G (RET)

Execution can be stopped in four ways:

1. A breakpoint. (Breakpoints are discussed in the next section.)
2. A user interruption via the BRK key (INTERRUPT key in executive Delta.)
3. An error causing a machine trap (illegal instruction, memory protection violation, etc.).
4. A normal program exit.

In each case the values of ;I, ;C, and ;F (See Table 24 on Special Symbols) are set, the cause of the stop is reported by an appropriate message, and terminal control returns to the user.

Example:

BRK AT .5C3
PRIVIL INSTR AT .77B
;I = .77B

Proceeding from a stop condition is accomplished by typing the ;P or ;G command. Execution continues from the location specified by the current value of ;I (i.e., where execution left off). The ;P command has an optional special format for use with instruction breakpoints. This is discussed in the section on breakpoints. For user interruptions via the

BRK key, the ;P and ;G commands cause execution to continue as if the interruption had not occurred.

```
BRK at .68C
;P (RET)
```

Proceeding from a machine trap causes reexecution of the violating instruction and another trap.

```
MEM PROTECT FAULT AT .74B
;P (RET)
MEM PROTECT FAULT AT .74B
```

The e;X command assembles and executes the expression e. The expression e must be an assembly-language instruction.

Examples:

```
LH,3  TABLE+4;X
STB,6  *LOC;X
```

If the expression does not result in a legitimate instruction, an error message is typed.

In most cases the instruction is executed and then terminal control returns to the user. However, if the expression is a branch instruction, control goes to the user's program (or causes a memory violation). Thus, the commands B GO;X and GO;G are equivalent. If the expression is a subroutine jump, the subroutine is entered. If the subroutine returns normally (i.e., to the calling location plus 1, 2, or 3), control returns to Delta and terminal control returns to the user. If the return is to other than the calling location plus 1, 2, or 3, the results are unpredictable.

The) command controls step mode execution. It executes the instruction in the currently open cell and opens and displays the next program step. If the instruction executed by) causes a branch, the effective branch address specifies the location to be opened and displayed. By using the / command to open and display a location and repeatedly issuing the) command, the user can proceed step-by-step through his program.

BREAKPOINTS: THE ;B, ;T, ;D, AND ;Y COMMANDS

Delta provides the user with multiple breakpoints of three types:

1. Instruction breakpoints.
2. Data breakpoints.
3. Transfer breakpoints.

The BRK key also causes a break in execution and is discussed in this section.

Eight instructions and eight data breakpoints are available to the user. Transfer breakpoints are limited only by options within the transfer breakpoint command.

As each breakpoint is reached, a small amount of information is printed out, giving the breakpoint location and an associated value. An optional "trace" mode allows execution to continue automatically after the breakpoint report to provide a flow-trace of both execution control and variation of data values.

INSTRUCTION BREAKPOINTS

Instruction breakpoints allow the user to halt execution at specified locations in the logical flow of his program. Eight instruction breakpoints, numbered 1 to 8 may be set. The command has the format

```
e[,n][,loc];B
```

where

- e specifies the location of an instruction. The breakpoint stop occurs just before execution of the instruction at e.
- n specifies the number of the breakpoint. If n is not specified, Delta assigns the next available breakpoint. If all instruction breakpoints are used, the error message NONE is typed. The user may then release one of the eight instruction breakpoints he has set and try again. (Releasing breakpoints will be discussed shortly.)
- loc specifies a location, the contents of which is to be displayed when the breakpoint is reached. Registers as well as core locations can be displayed.

The following list shows the format of the command when various parameters are omitted:

```
e;B
e,n;B
e,,loc;B
```

The breakpoint stop occurs just before execution of the instruction at e. When the breakpoint is reached, Delta prints the number and type of breakpoint, its location, and optionally the contents of the location specified by loc.

Examples:

```
A+3,1;B  A;G (RET)
1;B>A+.3
A+8,1,FF;B  ;G (RET)
1;B>A+.8  FF/  .54
```

When stopped at a breakpoint, the user may examine and modify his program as appropriate and then continue from the point of interruption by giving the command

```
;P
```

or

```
n;P
```

For instruction breakpoints, the ;G command will cause the break to reoccur and execution will not proceed. The ;P command bypasses the special breakpoint code at the point of interrupt and execution of the program can proceed. If the command n;P is given, program execution resumes as with the ;P command but the breakpoint that caused the interrupt will be passed n times before the break occurs again.

Example:

```
PH+8,2,R2;B   PH;G (RET)
2;B>PH+8 R2/   .4   ;P (RET)
2;B>PH+8 R2/   .5   ;P (RET)
2;B>PH+8 R2/   .6   5;P (RET)
2;B>PH+8 R2/   .12
```

(The breakpoint was passed five times before it caused this interrupt.)

If the user wishes to trace a particular instruction, he may give any of the four forms of the breakpoint command and specify the trace mode with a T following the B. That is,

```
e,n;BT
e;BT
e,n,val;BT
e,,val;BT
```

In this mode when the instruction e is reached, the breakpoint reporting information is printed and execution continues automatically.

Example:

```
A+3,4,5;BT   A;G (RET)
4;B>A+3  5/   54
4;B>A+3  5/   -1
4;B>A+3  5/   -175
```

The trace mode may be set after a breakpoint occurs with the ;T command, which sets the trace mode at the current breakpoint instruction.

Instruction breakpoints may be removed by

1. Giving an instruction breakpoint command that specifies the same breakpoint number as the instruction breakpoint to be removed.

   Example:

   ```
   AA,2;B (RET)
   .
   .
   .
   FF,2;B (RET)        (There is no longer a breakpoint
                        at AA.)
   ```

2. Giving the command n;B that specifies that the nth instruction breakpoint is to be removed.

3. Giving the command 0;B that specifies that all instruction breakpoints are to be removed.

The current instruction breakpoints may be listed for inspection with the ;B command. The list has the following form for each established breakpoint:

n[T]loc display

where

n is the breakpoint number

T indicates that the trace mode is set for that breakpoint.

loc is the breakpoint location.

display is the address to be displayed when the breakpoint occurs.

CALs, XPSDs, or LPSDs that depend on following calling sequences will not operate properly if they have an instruction breakpoint on them. BALs are not limited in this way.

DATA BREAKPOINTS

Data breakpoints allow the user to halt execution when a specified memory location changes value in a specified way. Eight data breakpoints (numbered 1 through 8) may be set. The command has the format

e[,n][,val][,m];D[r]

where

e specifies a memory location. When the contents of this location changes, a break will occur (unless other optionally specified requirements are not met).

n specifies the number of the breakpoint. If n is not specified, Delta assigns the next available breakpoint. If all data breakpoints are used, the error message NONE is typed. The user may then release one of the eight data breakpoints he has set and try again. (Releasing breakpoints will be discussed shortly.)

val specifies a value that is compared with the value in e. The parameters val and r must both be present if either one is present. The parameter val will be discussed further when r is discussed.

r specifies a relationship such as less than or equal to. When r and val are specified, a breakpoint will occur only whenever the contents of the memory location at e is in relation r to val. If no r and val specifications are given, a breakpoint occurs for all changes in the data and if a mask m is specified, it is ignored.

The letters used for r and their meanings are

LS	$(e)_c < val$	Contents of e under m is less than value.
EQ	$(e)_c = val$	Contents of e under m is equal to value.
GR	$(e)_c > val$	Contents of e under m is greater than value.
GQ	$(e)_c \geq val$	Contents of e under m is greater than or equal to value.
LQ	$(e)_c \leq val$	Contents of e under m is less than or equal to value.
NQ	$(e)_c \neq val$	Contents of e under m is not equal to value.

m specifies a mask. If m is specified, the contents of e are masked under m before being compared with val. The default mask is all one's.

Some specific variants of data breakpoint commands are given below.

e,n;D Sets data breakpoint n. Terminal control returns to the user after each change in the contents of e and printing of the data breakpoint message.

e;D Sets next available data breakpoint. If all data breakpoints are used, the error message NONE is typed. Terminal control returns to the user immediately after each change in the contents of e and printing of the data breakpoint message.

e,,val;Dr Sets next available data breakpoint with value, val, and relation, r. Terminal control returns to the user when the contents of e stand in relation r to the value val and the data breakpoint message has been printed.

e,,val,m;Dr Same as above except that the contents of e are masked by the mask m before being compared with val.

Some sample breakpoint settings are:

```
A,1,3;DGR
A+5,2,.FF,.FF;DEQ
AB,3;D
SDS,4,CSC;DGE
```

A T or trace parameter applies to all data breakpoint commands in the same way and with the same effects as described above for instruction breakpoints. For example,

```
A,1,3;DTGR
```

Also the command ;T may be given to set the trace mode at the current breakpoint (which just caused an interrupt.)

The output resulting from a data breakpoint has the form

```
n;D > loc e/cont
```

where

n is the number of the breakpoint.

loc is the location of the data modifying instruction.

e is the data address in question.

cont is the new value as just modified.

Example:

```
4;D > ADD   SUM/.14
```

When stopped at a data breakpoint, the user may examine and modify his program as appropriate and then continue from the point of interruption by giving the command

```
;G
```
or
```
;P
```
or
```
n;P
```

These commands are discussed in the previous section, "Instruction Breakpoints". (For data breakpoints, the ;G command is effectively the same as the ;P command.)

Data breakpoints may be removed by

1. Giving a data breakpoint command that specifies the same breakpoint number as the data breakpoint to be removed.
2. Giving the command n;D that specifies that the nth data breakpoint is to be removed.
3. Giving the command 0;D that specifies that all data breakpoints are to be removed.

The current data breakpoints may be listed for inspection with the command ;D. The list has the following form for each established breakpoint:

```
n[T]loc   cond   value   mask
```

where

n is the breakpoint number.

T indicates that the trace mode is set.

loc is the breakpoint location.

cond is the breakpoint condition relation.

value is the breakpoint value.

mask is the mask under which the data is tested.

The data breakpoint does not detect changes caused by direct hardware I/O transfers into the user's area nor does it detect changes in a temp stack caused by a push instruction (PSM, PSW). It does detect the change to the stack pointer doubleword.

TRANSFER BREAKPOINTS AND INTERPRETIVE EXECUTION

Transfer breakpoints allow the user to halt or trace execution when a branch instruction is encountered that branches when executed. This command differs from the other two breakpoint commands in that it initiates execution as soon as the command is decoded and processed. The format of the transfer breakpoint command is

[loc][,option$_1$][,option$_2$];Y

where

loc specifies a location at which to begin execution of the program. The default value is the value of the current location counter.

option$_1$ indicates whether or not an interrupt should be allowed to occur at the branches specified in the special action table (SAT) which will be described below. If option$_1$ = 0, then all branches except those specified in the SAT are to be processed as possible transfer breakpoints. If option$_1$ = 1, then only those branches specified in the SAT are to be processed as possible transfer breakpoints. If this option is omitted and the SAT contains no entries, then all branches are processed as possible transfer breakpoints. If the option is omitted and the SAT does contain entries, then the default value for the option is zero (so that all branches except those specified in the SAT are to be processed as possible transfer breakpoints).

option$_2$ indicates whether or not BDR and BIR branches are to be processed as possible transfer breakpoints. If option$_2$ = 0, then BDR/BIR branches are not to be processed as possible transfer breakpoints. If option$_2$ = 1, then BDR/BIR branches are to be processed as possible transfer breakpoints. The default value is 0.

The following list shows the format of the command when various parameters are omitted:

;Y

loc;Y

loc,option$_1$;Y

loc,,option$_2$;Y

,option$_1$;Y

,,option$_2$;Y

,option$_1$,option$_2$;Y

When a break occurs as the result of the transfer breakpoint command, the following message is output:

loc1 ⟶ loc2

where

loc1 is the address of the branch instruction that just branched.

loc2 is the address of the instruction to which the program branched.

Execution may be continued with the ;P or ;G command. The ;P command with a proceed count (n;P) is not meaningful in the transfer breakpoint mode.

If the user wishes to use the trace mode with the transfer breakpoint command, he may give any of the forms of the command and specify the trace mode with a T following the Y. For example:

;YT

loc,,option$_2$;YT

loc,option$_1$,option$_2$;YT

In this mode, when a breakpoint occurs, the breakpoint reporting information is printed and execution continues automatically.

The trace mode for all transfer breakpoints may be set after a transfer breakpoint break occurs. The command which sets the trace mode is ;T.

The transfer breakpoint mode may be turned off with the command:

0;Y

Special Action Table (SAT). The special action table lists up to eight locations in the user's program. These locations are meaningful only if they contain branch type instructions. The action to be taken depends on option$_1$ of the transfer breakpoint command.

The following command enables the user to set entries in the SAT:

loc1[,loc2[,loc3[,loc4]]];YS

The command enters the specified locations in the SAT if space is available.

The command

loc1[,loc2[,loc3[,loc4]]];YR

releases specified locations from the SAT.

The command ;YR releases all SAT entries. The command ;YD displays the SAT.

BRK KEY BREAKPOINTS

At any time during program execution the user may halt his program by pressing the BRK key. A message is printed for the user, giving the location of the breakpoint. If the user hits the BRK key while his program is in execution, the message is:

BRK AT loc

After such a breakpoint, the ;P or ;G command continues execution.

If the breakpoint occurs while Delta is executing, the message is

BRK IN DELTA

The user may then give any of the Delta commands.

MEMORY SEARCH AND MODIFICATION: THE ;W, ;N, ;M, AND ;L COMMANDS

There are two search commands, e;W and e;N. The e;W command searches for values which match the expression e and displays the location and contents of each cell containing the value. The e;N command searches for cells that do not contain the expression e and displays their location and contents.

The search is carried out between the limits determined by the symbol table values of ;1 and ;2. The special symbols ;1 and ;2 identify the lower and upper search bounds respectively. The initial value of ;1 is the lowest current user data area address, and the initial value of ;2 is the highest current user data area address. Usually the initial value of ;2 is greater than the last address of the user's program and this causes a trap to occur when a search is requested. Therefore, the user should always set limits on the area in which the search is to be done by using the e;1 and e;2 commands. The field e is an expression which specifies the bound location. An example is given below:

AA;1 EE;2 'ABCD';W (RET)

In the example, each cell between AA and EE will be searched for the EBCDIC value ABCD. The location and contents of each cell containing that value will be displayed.

Both bounds may be set by one command, the ;L command. The format of the ;L command is

e1,e2;L

where e1 specifies a value for ;1 and e2 specifies a value for ;2. The example above might also be written

AA,EE;L 'ABCD';W (RET)

When the user sets the search bounds, they remain set at the specified value until they are reset by the user. The bounds do not revert back to their initial values after a search has been performed.

The search may examine entire cells or portions of cells. This is determined by a mask which is identified by the special symbol ;M. The initial value of ;M is all ones, so that entire cells will be examined. The mask ;M may be reset by the ;M command which has the format

e;M

The expression e is used to set the bits of ;M to a particular pattern of ones and zeros. Only those bits corresponding to the one bits of the mask will be examined when a search is performed. For example:

.FF000000;M

The mask will be set so that only the first eight bits of each cell will be examined to see if they match the value being searched for.

Like the search bounds, the value of the mask ;M will not be changed until another ;M command is given.

In the following example, only the last byte of the cells AA through EE will be examined. Those containing the EBCDIC value 'D' will be displayed.

AA,EE;L .000000FF;M 'D';W (RET)

(In the .000000FF;M command, the leading zeros are not required.)

The user may express values to be searched for in their assembly-language format or in their machine-language format. In the example below, all words between ABC and ABC+.100 with the last 17 bits equal to the address of the ERR will be displayed as shown.

```
.1FFFF;M    ABC,ABC+.100;L    ERR;W (RET)
ABC+.3/     BAL,4     ERR
ABC+.A/     BAL,4     ERR
ABC+.D/     BAL,4     ERR
ABC+.6A/    AWM,1     ERR
```

A second value may be specified in the ;W and ;N commands so that the formats of the commands are

e1,e2;W and e1,e2;N

The e2 field specifies a value which will be stored through the mask ;M into all locations that meet the specified condition (i.e., match or mismatch). Locations meeting the conditions will be displayed after the substitution has taken place. The following example is the same as the example above, except that the symbol OUT will be substituted for ERR. (OUT must be a defined symbol within the program.)

Example:

```
.1FFFF;M    ABC,ABC+.100;L    ERR,OUT;W (RET)
ABC+.3/     BAL,4     OUT
ABC+.A/     BAL,4     OUT
ABC+.D/     BAL,4     OUT
ABC+.6A/    AWM,1     OUT
```

The user may interrupt an in-progress search by pressing the BRK key. Delta halts the search and returns terminal control to the user.

MEMORY CLEARING: THE ;Z COMMAND

The ;Z command is basically used to clear (i.e., set to zeros) specified areas of memory. The basic format of the ;Z command is

```
e1,e2;Z
```

where expression e1 is the lower limit and expression e2 is the upper limit of the memory area to be cleared. An error results if the value of e2 is less than that of e1. Also e1 and e2 must not specify addresses outside of the user's area in memory.

A third field may be added to the ;Z command so that the format is

```
e1,e2,v;Z
```

The field v specifies a value to be stored into each of the memory cells in the area delimited by e1 and e2. In this way, the ;Z command may be used for purposes other than clearing memory.

Examples:

```
A,A+5;Z (RET)
.1CE0,.1CF0;Z (RET)
ALPHA,ALPHA+2,1;Z (RET)   (Stores the value 1 into the three memory cells ALPHA, ALPHA+1, and ALPHA+2).
```

DISPLAY MODES: THE ;A, ;R, AND ;RK COMMANDS

The ;R and ;A commands control the way in which Delta displays location values when typing the contents of cells. The mode display is either relative (;R) or absolute (;A). When in the relative mode, Delta looks up location values in the symbol table and displays the symbol if one corresponds exactly to the value. If no exact correspondence is found, Delta displays the symbol with the next smaller value followed by a word offset in hexadecimal. If the mode is absolute (;A), then location values are displayed as hexadecimal numbers. Note that these commands control the display of location values but not the display of the address portion of instructions contained in those locations. Examples of the ;R and ;A commands are shown below:

;R Display Example:

```
A,A+5/    LI,1     .10
A+.1/     CW,1     K45
A+.2/     BGE      ZZZ
A+.3/     AI,1     1
A+.4/     B        A17
ZZZ/      STW,2    BR13
```

;A Display Example:

```
A,A+5/    LI,1     .10
.5CD/     CW,1     K45
.5CE/     BGE      ZZZ
.5CF/     AI,1     1
.5D0/     B        A17
.5D1/     STW,2    BR13
```

The ;R command may be preceded by a value (n;R) that sets the maximum offset to be used in address output. If no symbol lies within "offset" of the value, the address is printed as absolute hexadecimal. Thus, 10;R causes Delta to display symbol plus relative offset only when a symbol lies within 10 locations of the display address.

The ;RK command sets relative address output mode, using only control section type symbols for output unless there is an exact match between the symbol value and output value (for a discussion of setting the control section type, see "Symbol Table Control: The ;U, ;K, ;S, !, and <> Commands" earlier in this chapter). If there are no control section symbols, the output is hexadecimal. Thus, output is "control section plus hexadecimal offset", "symbol", or "hexadecimal constant".

;RK Display Example:

```
A,A+5/    LI,1     10
.5CD/     CW,1     K45
.5CE/     BGE      ZZZ
.5CF/     AI,1     1
.5D0/     B        A17
ZZZ/      STW,2    BR13
```

PRINTER OUTPUT: THE ;O AND ;J COMMANDS

These two commands provide for output (via symbionts) to the line printer. The ;O command produces hexadecimal dumps on the line printer, while the ;J command directs all Delta output to the line printer. This is particularly

useful in the cases of large formatted displays and output from tracing breakpoints.

The printer and tape I/O routines are completely self-contained in the executive version with no dependence on system I/O routines. The executive version of Delta operates with all interrupts except console interrupts disabled. Examples of the ;O and ;J commands are

e1,e2;O [header] contents of memory from location e1 through location e2 are printed on the line printer, single-spaced, eight hexadecimal words with initial hexadecimal location value per line. Duplicate lines are suppressed. If any input follows the O, it is printed as a header. Each dump begins at the top of a fresh page with the contents of the general registers printed first.

;J toggles the output location switch that alternates between the terminal and the line printer each time the command is given. Output from the equal command, from nontracing breaks, from trap, abort, and error returns, and from syntax and other error conditions in Delta are always directed to the terminal. Examples are

```
A, 1;B
X,2,3;DLS    ;J    B;G
```

Note: The output that would have appeared here from data break 2 goes to the line printer.

EXECUTIVE DELTA

Executive Delta does not honor the following commands:

;Y

;S

;R

All other Delta commands may be used. Special executive Delta restrictions have been noted throughout this chapter.

WRITING PROGRAMS WITH DELTA

The user may write and check short Meta-Symbol or machine language programs using Delta. The following two commands are especially helpful for writing programs:

1. Symbolic tags may be defined at a specific address using the command

   e(f<s>[K]

   (See the section "Symbol Table Control".) Each symbolic tag should be defined with this instruction before it is used in the program. The range of addresses available to the user is .C000-.1BFFF.

2. Pages for the program may be requested from the Monitor by using the command

   e\

   (See the section "Memory Cell Opening and Display".) This command also opens the specified cell so that the user may store an instruction or data into it.

   Example:

   .10000(R<BEGIN>K ⏎ defines the tag BEGIN at location .10000.

   BEGIN LI,2 0 ⏎ opens the cell at BEGIN and requests the page from the Monitor (if it is not already assigned to the user). The instruction LI,2 0 is then stored into the cell at BEGIN.

ERRORS AND ERROR MESSAGES

Errors that result in machine traps are reported to the user, and console control is returned to the user to await further commands. Each message is accompanied by the location, symbolically if possible, of the offending instruction. The messages are

NONEXIST INSTR AT

NONEXIST MEM REF AT

PRIVIL INSTR AT

MEM PROTECT FAULT AT

STACK LIMIT FAULT AT

UNIMP INSTR AT

FIXED ARITH OVFLW AT

FLOAT FAULT AT

DECIMAL FAULT AT

Syntax errors are reported by the message "?n", where n is the number of the character in the command line that Delta was processing when the error was detected. This message is sent to the user whenever Delta cannot understand the user's command syntax. Because the commands are brief (i.e., requiring few keystrokes) and most errors can be spotted easily by eye, only a few syntax errors are explicitly commented. Example errors and Delta's response to them are listed below:

'ABCDE'= ? 6	Constant value larger than one word.
ABC;K ⏎ ? 5	Symbol not in symbol table.
FF;M 100,XY;L .6B;W ⏎ ?13	Symbol value not found. Remainder of command string ignored.

A,5;E (RET) ? 5	Command unknown.
LW*5 ALPHA= ? 3	Asterisk in the wrong place.
.3ACR/ ? 5	Illegal character in a hexadecimal number.
(B;/ ? 2	Illegal format control character.
LOC,,3;DNE (RET) ? 10	Illegal relation.
;T (RET) ? 2	No break in an attempt to set trace mode on.

PROGRAM EXITS

When called, Delta takes control of program exits via the CAL M:SXC. Delta reports execution of exit CALs with a message of the form

EXIT n AT loc

where

n is the exit code as defined in the table below.

loc is the address of the CAL or instruction causing exit.

Code	Type of Exit	Example
0	Normal	M:EXIT.
1	Trap error	Decimal or floating trap.
2	I/O error	No error address.
4	Limits	Maximum time; maximum pages output.
10	Termination	Operator aborted job.
20	Termination	Operator errored job.
40	Abnormal	M:XXX.
80	Job errored	M:ERR.

DELTA COMMAND SUMMARY

The Delta commands are summarized in Table 25. They are listed by groups according to the types of function they perform.

Table 25. Delta Command Summary

Command	Function
Expression Evaluation	
e=	Evaluates and types the value of the expression e in the most appropriate format.
e(f=	Evaluates and types the value of e in format f.
=	Following a display, evaluates and types the value of the last expression typed by Delta.
(f;=	Changes the default format for output for the = command to the format specified by f.
Memory Cell Opening and Display	
e/	Displays the contents of a cell e in the most appropriate format, and opens the cell in preparation for change.
e(f/	Opens and displays the contents of cell e in format f.
e1,e2/ e1,e2(f/	Displays the contents of cell e1 through e2 in the most appropriate format or in the specified format f, and opens cell e2.
e\	Opens but does not display cell e. Also may be used to request pages from the Monitor. (The \ command is replaced by ¢ in the executive version.)
/	Following a display, displays but does not open the last cell addressed by the display. The new display is in the default format.

Table 25. Delta Command Summary (cont.)

Command	Function
Memory Cell Opening and Display (cont.)	
(TAB)	Following a display, displays and opens the last cell addressed by the display.
(f;/	Changes the default format for output for the slash command to the format specified by f.
Memory Modification	
e (RET)	Stores the word specified by e in the currently open cell and closes the cell.
e (LF)	Stores e in the currently open cell, closes it, and opens and displays the next higher addressed cell. (The LF is replaced by EOM in the executive version.)
e↑	Stores e in the currently open cell, closes it, and opens and displays the next lower addressed cell. (The ↑ is replaced by & in the executive version.)
e (TAB)	Displays and opens the cell addressed by the last quantity typed. If an expression precedes the TAB, the expression is stored in the open cell and that cell is closed.
Symbol Table Control	
s;S	Selects internal symbol table s.
;S	Loads global symbol table.
;U	Displays undefined symbols.
e(f<s>[K]	Assigns to symbol s the value e and the format f.
s(f!	Assigns to symbol s the value of the currently open cell and the format code f.
s;K	Flags symbol s in the symbol table. It will not be used in output expressions, but it can still be used in input expressions.
;K	Removes all symbols except instruction mnemonics and special symbols.
;KI	Removes the current internal symbol table.
;KG	Removes the global symbol table and any symbols defined from the console.
Execution Control	
e;G	Begins execution at e.
;G	Begins execution at the address specified by the current location counter value.
;P	Begins execution at the address specified by the current location counter value.
n;P	Proceeds with no output the next n times the current instruction breakpoint is encountered.
e;X	Executes the instruction e.
)	Executes the current instruction and displays the next one.

Table 25. Delta Command Summary (cont.)

Command	Function
	BREAKPOINTS
Instruction Breakpoints	
e,n;B	Sets the nth instruction breakpoint at location e.
e,n;BT	Same as above, but the program automatically proceeds from the breakpoint after the breakpoint message is printed (trace mode).
e;B	Sets the next available breakpoint at location e.
e;BT	Same as above, but the program automatically proceeds from the breakpoint after the breakpoint message is printed (trace mode).
e,n,loc;B	Sets the nth instruction breakpoint at location e and causes the contents of loc to be displayed when the break occurs.
e,n,loc;BT	Same as above, but the program automatically proceeds from the breakpoint after the breakpoint message is printed (trace mode).
e,,loc;B	Sets the next available breakpoint at location e and causes the contents of loc to be displayed when the break occurs.
e,,loc;BT	Same as above, but the program automatically proceeds from the breakpoint after the breakpoint message is printed (trace mode).
;T	Sets the trace mode at the current breakpoint (which just caused a breakpoint interrupt).
n;B	Removes the nth instruction breakpoint.
0;B	Removes all instruction breakpoints.
;B	Displays all active instruction breakpoints.
Data Breakpoints	
e,n,val,m;Dr	Causes data break n to occur whenever the contents of cell e, masked by m, are in relation r to val. The relations are LS e < val EQ e = val GR e > val GQ e ≥ val NQ e ≠ val LQ e ≤ val
e,,val,m;Dr	Same as above, but uses the next available data breakpoint number.
e,n,val,m;DTr e,,val,m;DTr	Same as the two above, but the program automatically proceeds from the breakpoint after the breakpoint message is printed (trace mode).
e,n;D	Causes data breakpoint n to occur whenever the contents of cell e are changed.
e;D	Same as above, but uses the next available data breakpoint number.
e,n;DT e;DT	Same as the two above, but the program automatically proceeds from the breakpoint after the breakpoint message is printed (trace mode).
e,,val;Dr	Sets the next available data breakpoint. A break will occur whenever the contents of e are in relation r to val.
e,,val,m;Dr	Same as above except that the contents of e are masked by the mask m.
e,,val;DTr e,,val,m;DTr	Same as the two above, but the program automatically proceeds from the breakpoint after the breakpoint message is printed (trace mode).

Table 25. Delta Command Summary (cont.)

Command	Function
Data Breakpoints (cont.)	
;T	Sets the trace mode at the current breakpoint (which just caused a breakpoint interrupt).
n;D	Removes the nth data breakpoint.
0;D	Removes all data breakpoints.
;D	Displays all active data breakpoints.
Transfer Breakpoints and Interpretive Execution	
;Y	Starts execution at the current location counter in the transfer breakpoint mode. Does not display branches specified in the SAT. Does not display BDR and BIR branches.
loc;Y	Same as above except that execution begins at loc.
;YT loc;YT	Same as the two above, except that the trace mode is also set.
,option1;Y	Starts execution at the current location counter in the transfer breakpoint mode. Does not display branches specified in the SAT if option1 = 0. Displays only those branches specified in the SAT if option1 = 1. Does not display BDR and BIR branches.
loc,option1;Y	Same as above except that execution begins at loc.
,option1;YT loc,option1;YT	Same as the two above, except that the trace mode is also set.
,,option2;Y	Starts execution at the current location counter in the transfer breakpoint mode. Does not display branches specified in the SAT. Displays BDR and BIR branches if option2 = 1. Does not display BDR and BIR branches if option2 = 0.
loc,,option2;Y	Same as above except that execution begins at loc.
,,option,2;YT loc,,option2;YT	Same as the two above, except that the trace mode is also set.
,option1,option2;Y	Starts execution at the current location counter in the transfer breakpoint mode. Does not display branches specified in the SAT if option1 = 0. Displays only those branches specified in the SAT if option1 = 1. Displays BDR and BIR branches if option2 = 1. Does not display BDR and BIR branches if option2 = 0.
loc,option1,option2;Y	Same as above except that execution begins at loc.
,option1,option2;YT loc,option1,option2;YT	Same as the two above, except that the trace mode is also set.
;T	Sets the trace mode for all transfer breakpoints.
0;Y	Turns off the transfer breakpoint mode.
loc1[,loc2[,loc3[,loc4]]];YS	Sets one to four entries in the SAT (Special Action Table).
loc1[,loc2[,loc3[,loc4]]];YR	Releases one to four entries in the SAT.
;YR	Releases all entries in the SAT.
;YD	Displays the SAT.

Table 25. Delta Command Summary (cont.)

Command	Function
Memory Search and Modification Memory between the bounds specified in ;1 and ;2 (initially set to the lower and upper limits of memory assigned for user data) is searched under the mask in ;M (initially all ones). If field e2 is specified in the search command, the value in that field is stored through mask ;M into each location that meets the specified condition.	
e;W	Searches for and displays words that match e under the mask ;M.
e1,e2;W	Stores e2 through mask ;M in locations that match e1 through the mask.
e;N	Searches for and displays words that do not match e.
e1,e2;N	Stores e2 through mask ;M in locations that do not match e1 through the mask.
e;1	Sets the memory search lower bound to e.
e;2	Sets the memory search upper bound to e.
e1,e2;L	Sets ;1 to e1 and ;2 to e2.
e;M	Sets the search mask to e.
Memory Clearing	
e1,e2;Z	Zeros memory from e1 through e2.
e1,e2,v;Z	Stores the value v in memory from e1 through e2.
Display Modes	
;R	Sets the display mode in memory addresses to symbol plus relative hexadecimal offset.
n;R	Same as above, but sets the maximum hexadecimal offset to n.
;RK	Displays addresses as control section type symbol plus any hexadecimal offset. If the value displayed is equal to that of any symbol, then the symbol is displayed. If there is no control section type symbol, then a hexadecimal constant is displayed.
;A	Sets the display mode for locations to hexadecimal numbers.
Printer Output	
e1,e2;O [header]	Prints the contents of memory from location e1 through location e2 on the line printer in the standard core memory dump format. If any input follows the O, it is printed as a header.
;J	Toggles the output location switch which alternates between the terminal and the line printer each time the command is given.

8. LINK PROCESSOR

INTRODUCTION

The on-line linking and loading of programs is carried out by the Link processor. Link constructs a single entity called a load module (LM) which is an executable program formed from relocatable object modules (ROMs). Link also provides the necessary data space and program linkages for the association of public libraries.

Link is a one-pass linking loader that makes full use of mapping hardware. It is not an overlay loader. If the need for overlays exists, the overlay loader must be called by entering the job in the batch stream (see UTS/BP Reference Manual, 90 17 64).

The access protection types provided by Sigma 6,7, or 9 hardware are

00 read, write, and execute access permitted (data)

01 read and execute access permitted (pure procedure)

02 read access permitted (static data)

03 no read or write permitted (no access)

The final program resulting from a linking operation has three protection types, one for data, one for pure procedure, and one for DCBs. Static data and nonaccess information, if specified, are loaded with the pure procedure.

LOAD MODULE STRUCTURE

A load module formed by Link is composed of three parts: program, global symbol table, and internal symbol table. Each of these parts is described in the following sections.

PROGRAM

A program may be sectioned into six parts: pure procedure, data, common, DCBs, public libraries, system library.

1. Pure Procedure

   This section of code contains machine instructions and is generated by compilers and assemblers with protection type 01 (read and execute access). Sections with a nondata protection type (static data and no access) are also included here.

2. Data or Program Context

   This section is generated by the compilers and assemblers with protection type 00 (read, write, and execute access).

3. Common

   This is blank common storage is generated by compilers and assemblers as a dummy section with the name F4:COM. The size of blank common storage is determined by the first size declared. All subsequent F4:COM declarations must be less than or equal to that size.

4. DCBs

   A data control block (DCB) is a table containing the information used by the Monitor in performance of an I/O operation. At the end of a link operation, Link constructs a DCB corresponding to each outstanding external reference with names beginning with F: and M:.

   The M:UC DCB, which is the DCB most commonly used for terminal I/O, is supplied as a portion of the user's JIT (job information table); any M:UC reference is automatically satisfied thereby. The default assignment of M:UC to the user's terminal is unalterable. (Output operations via M:UC are treated specially by the Monitor; see Chapter 10.) If the program being linked does not contain a reference to M:DO, a reference to it is supplied by Link, since diagnostic output is generally written via this DCB. If the user does not want this DCB to be constructed, due to space considerations, he can explicitly reference M:DO and satisfy the reference (vacuously) within his program. (Some diagnostic output is likely to be lost.)

   A DCB name of the form M:ab, where ab corresponds to an operational label, is considered a reference to a standard system DCB. The standard system DCBs are discussed in terms of operational labels and default assignments in UTS/BP Reference Manual, 90 17 64.

   DCBs constructed by Link are 51 words long and consist of

   a. A 22-word standard initial segment, containing a standard default operational label if the DCB is one of the system DCBs.

   b. Five variable length items including a control word for each, with space for

      - A three-word file name.
      - A two-word account number.
      - A two-word password.
      - A three-word block for three input serial numbers.
      - A three-word block for three output serial numbers.
      - A two-word block for expiration date.

   c. An eight-word key buffer.

   The standard system DCBs also exist in ROM form on files in the system account; in this form they differ from Link-constructed DCBs in size and composition, as described in UTS/BP Reference Manual, 90 17 64. These ROMs can be explicitly named in a LINK or RUN command to satisfy corresponding references.

While allocating, constructing, and combining DCBs, Link guarantees that each DCB is contained within a page. This allows the operating system to access DCBs in either mapped or unmapped mode. User-supplied DCBs (i.e., DSECTs with names beginning M: or F:) are placed in the DCB record, in user-context space, together with those constructed by Link. All are given protection type 02.

5. Public Libraries

Any UTS installation can define a set of subroutines that constitute a public library. The installation may specify several different public libraries containing collections of routines that are useful in various environments. Only one library may be associated with an executing program. DEF stacks for public libraries are stored under special names in the system account and are used to link programs to them. See the UTS/SM Reference Manual, 90 16 74, Chapter 6, for more detailed information on the structure and creation of public libraries.

Only one block of core memory is required for the public library no matter how many users are using it. However, use of just one routine in the public library requires core for the entire package. The reentrant portion of each library is shared among users (on-line and batch), thus saving physical core memory and allowing for more efficient system operation. User-dependent data storage for each library routine is allocated by Link at a fixed virtual address. Thus, each public library is constructed in two parts: reentrant procedure and direct access data. By forming the library in this manner, a speed advantage of from 5 to 20 percent over push-down storage reentrancy is obtained.

UTS provides three public libraries: P0, P1, and J0 (only the first two are of general interest). Library P1 contains the most commonly required routines from the Extended FORTRAN IV run-time and mathematical library (about 65 routines). Library P0 includes library P1 plus the FORTRAN Debug Package (FDP). These two libraries will satisfy the requirements of the majority of users for program execution and debugging, respectively. (The remainder of the run-time and mathematical routines comprising the entire Extended FORTRAN IV subprogram library reside on the system library, described below.) Public library J0 contains the user-JIT Definition Package. (See Chapter 6 of the UTS/SM Reference Manual, 90 16 74, also for more detailed descriptions of libraries P0, P1, and J0.) Additional public libraries created by a user-installation may be named P2-P9.

6. System Library

The system library consists of approximately 170 FORTRAN IV library routines in ROM form, on file :BLIB in the :SYS account. Searching of this library is implied by the default library-search code L in a LINK or RUN command. This library is always searched last if any unsatisfied references remain unless the NL option is specified. Routines that are obtained from the system library become part of the user program and are not shared. Thus, core is required for each system library routine. The speed advantage is still maintained since each routine includes any necessary data.

GLOBOL SYMBOLS

While performing the linking process, Link constructs a global symbol table. This table is a list of correspondences between symbolic identifiers (labels) used in the original source program and the values or virtual core addresses that have been assigned to them by Link. The global symbols define (DEF) objects within a module that may be referenced (REF) in other modules. This table is available to Delta for use in debugging.

INTERNAL SYMBOLS

An internal symbol table is a list of correspondences similar to the global symbol table but applies only to symbols defined within the module. Each internal symbol table constructed by Link is associated with a specific input file and is identified by its name. This table is also available to Delta for debugging.

When an internal symbol is equated to an external symbol with an addend, and the module containing the external definition is in a different file from the module containing the external reference, the file containing the definition must appear on the LINK or RUN command before the file containing the external reference. Furthermore, an internal symbol should not be equated to an external reference with an addend satisfied from a library.

No internal symbol table is generated for a named library (one with a fid).

SYMBOL TABLES

Delta makes it possible to reference both global and internal symbols at the time programs are debugged. Programs formed by loaders, together with the tables of global and internal symbols, are operated on in a code similar to assembly language symbolic code.

Global and internal symbol tables, as formed by Link and used by Delta, consist of three word entries. Symbolic identifiers (labels) are limited to seven characters. Symbols originally longer than seven are truncated, leaving the initial seven characters, although the original count is retained. Thus, symbols that are identical in their first seven characters and are of equal length occupy one position in the symbol table. The value retained for multi-defined symbols is the first one encountered during the linking process. Each symbol entered into the table has an internal resolution and a type classification. Internal resolutions are: byte, halfword, word, doubleword, and constant. Symbol types are: instruction, integer, EBCDIC text, short floating-point, long floating-point, decimal, packed decimal, and hexadecimal.

Object language code produced by UTS assemblers and compilers provides internal symbols with internal resolution and type classification. UTS loaders retain of this information in processing object language code.

CONVENTIONS

The terminal and language conventions for Link are the same as for TEL except the function of the BREAK key. If the BREAK key is depressed while a LINK command is being entered, the command is ignored and a new command must be typed (as if X^C had been pressed).

LINK COMMANDS

The Link processor is called implicitly by a LINK or RUN command given at TEL level, as described in Chapter 3.

Example:

Assume there are two relocatable object modules. The internal symbols for the first module (MFL1) are to be left out of the resulting load module, but the internal symbols for the second module (MFL2) are to be included. The resulting load module is called LM1.

```
! LINK (NI) MFL1,(I) MFL2 ON LM1 (RET)
!
```

If Link needs additional information, it will identify the problem, and then prompt (:) for input.

Example:

Assume the same example as above except that Link cannot find MFL2 because it was supposed to be MFL3.

```
! LINK (NI) MFL1,(I) MFL2 ON LM1 (RET)
CANT FIND: RETYPE MFL2
: MFL3 (RET)
!
```

Note that the ROM specification indicated as unfound (e.g., MFL2) can, alternatively, be bypassed by responding with carriage-return only.

Example:

Assume that modules A and B are to be linked, with merging of internal symbol tables, to form output module C. In the linking process, one double definition (Z) and one internal unsatisfied definition (Y) are found.

```
! LINK (A,B) ON C (RET)
        LINKING A
        LINKING B
IDDEF       Z (internal double definition)
IUSAT       Y (internal unsatisfied reference)
```

ERROR MESSAGES

Whenever an error occurs during a linking operation, Link sends an error message to the terminal. Some of these messages are for syntax errors, others are for errors arising out of the linking operation. They are listed in Table 26. Most of these errors terminate the linking operation prematurely.

LINK COMMAND SUMMARY

Table 27 is a summary of the LINK and RUN commands. The left-hand column gives the command format, the right-hand columns gives the command function and options. Note that the format of the two commands differ only in the UNDER-clause options.

Table 26. Link Error Messages

Message	Description
CANT FIND :RETYPE rom	The specified relocatable object module cannot be found.
CARD CKS/COMPUTED CKS/cd/cp/	This message is sent to the terminal along with the CHECKSUM ERROR message. It specifies the card checksum (cd) and the computed checksum (cp).
CHECKSUM ERROR	A checksum error has occurred. The CARD CKS/ COMPUTED CKS/cd/cp/ message specifies the difference.
CORE LIBRARY OVERLAPS PURE PROCEDURE	There is insufficient virtual memory to contain the pure procedure and the core library REF/DEF stack.

Table 26. Link Error Messages (cont.)

Message	Description
DUMMY SECTION LARGER THAN PREVIOUS DEF	The dummy section initially defined was not the largest dummy section.
GLOBAL SYMBOL TABLE OVERLAPS PURE PROCEDURE	There is insufficient virtual memory to contain the pure procedure and the symbol tables.
ILLEGAL DATA FORMAT	Input modules did not contain ROM data.
ILLEGAL LOAD ADDRESS	An attempt was made to load outside the limits of the program.
ILLEGAL LOAD ITEM TYPE	ROM input data is illegal (e.g., it is load module data instead).
INSUFFICIENT PHYSICAL MEMORY TO CONTINUE	A request for a memory page has been refused.
I/O ERROR LINKING SYSTEM LIBRARY	This message usually indicates there is no system library.
I/O ERROR OPENING OUTPUT FILE	An I/O error occurred during the opening of an output file.
I/O ERROR READING ASSIGN MERGE RECORD	This message usually indicates there is no assign-merge record.
I/O ERROR READING CORE LIBRARY	This message usually indicates there is no core library.
MODULE #/SEQUENCE#/md/sq/	This message accompanies most other messages. It identifies the module number (md) and sequence number (sq) of the last card before the error. Both numbers start at zero.
MORE THAN 2 PAGES REQUESTED FOR DCBs	This message indicates that the limit of two pages for DCBs has been exceeded.
NO PROGRAM START ADDRESS	The program has no start address.
ON FILE fid ILLEGAL	ON was specified and the output file (fid) already exists.
SEQUENCE ERROR	A sequence error has occurred.
STACK OVERFLOW	An internal storage overflow has occurred.
UNEXPECTED END OF ROM DATA	EOF encountered before last card of ROM.
Note: All errors, except CANT FIND, cause abnormal termination of Link.	

Table 27. Link Command Summary

Command	Description
LINK [codes]rom[,rom]... [,rom] [ON / OVER lmn] [;lid ... [,lid]... [,lid]] [UNDER FDP]	Forms the load module as specified. Options (codes): library search: (L) search system library (NL) do not search system library default: (L) (Ji) or (Pi) associate ith public library where i = 0-9 (FDP) associated public library P0 (NP) do not associate any public library default: P1 display: (D) display undefined internal and external symbols (ND) do not display undefined internal and external symbols (C) display conflicting internal and external symbols (NC) do not display conflicting internal and external symbols (M) display load map (NM) do not display load map default: (D), (C), (NM) Options (symbol table): (I) include symbol table with LM (NI) do not include symbol table with LM default: (I) rom may be fid or $; parentheses enclosing mfl's cause merge of symbol tables. lid must name a file containing one or more ROMs.
RUN [codes]rom[,rom]... [,rom] [ON / OVER lmn] [;lid ... [,lid]... [,lid]] [UNDER DELTA / FDP]	Loads a specified load module and starts execution. Options: (see LINK command)

9. MONITOR SERVICES TO USER PROGRAMS

INTRODUCTION

All Monitor services available to UTS batch programs are described in the UTS/BP Reference Manual, 90 17 64. Those services that are unique to time-sharing are discussed in this chapter. In addition, a description of differences between on-line and batch responses to certain procedures is provided.

ON-LINE UTS SERVICE CALLS

SET PROMPT CHARACTER

M:PC Ordinarily, when control is turned over to an on-line program, a null prompt character is assigned. The PC routine allows an on-line program to set a prompt character. This character, if non-null, is typed (usually at the left margin) whenever input is requested from the terminal (UC device). If M:PC is used in a batch program, it is ignored.

The procedure call is of the form

```
M:PC    'character'
```

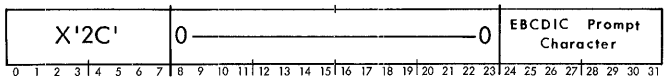
where character specifies the EBCDIC prompt character that is to be associated with the user program (an EBCDIC 00 or null character means that no prompt character is desired.)

Illegal EBCDIC characters and lower case ANSCII characters are not allowed. If there is an illegal character, no change in the prompt character will be made and CC1 will be set on return.

Calls generated by the M:PC procedure have the form

```
CAL1,1    fpt
```

where fpt points to the FPT shown below.



CHANGE TERMINAL TYPE

M:CT The CT routine allows an on-line program to switch among the terminal translations provided by the COC I/O routines. Tables related to each terminal type control the translation of characters transferred between the computer and the terminal.

The CT routine also affects other functions treated differentially by terminal type. These functions include certain line editing and terminal control functions.

The procedure call is of the form

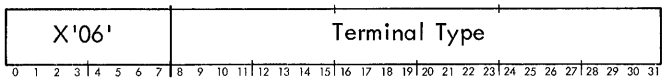
```
M:CT    number
```

where number specifies the number of the desired table (the range of the number is currently $0 \leq$ number ≤ 11.)

Calls generated by the M:CT procedure have the form

```
CAL1,8    fpt
```

where fpt points to the FPT shown below.



The current tables translate for Models 33, 35, and 37 Teletypes and the XDS 7015 Keyboard/Printer.

The assignment of terminal type numbers are

Number	Meaning
0	Teletype Model 33.
1	Teletype Model 35.
2	Teletype Model 37.
3	XDS Model 7015 Keyboard/Printer
4	IBM 2741 Terminal EBCD Standard.
6	IBM 2741 Terminal EBCD APL.
8	IBM 2741 Terminal Selectric Standard.
10	IBM 2741 Terminal Selectric APL.

CC1 is set if there is an illegal type code or M:CT is not in an on-line program.

CHANGE ACTIVATION CHARACTERS

A variation of the call corresponding to the M:CT procedure allows the calling program to choose among three sets of message-terminating, or activation, characters for terminal input. The normal set of activation characters is: CR, LF, FF, FS, RS, US, GS, EOT, SUB, and ESC F. Two additional activation sets are available that augment the normal activation set. They are:

1. "All" special graphics and control characters.

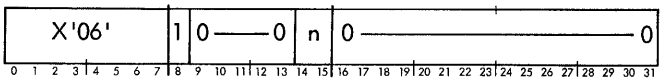
2. "All" control characters.

Character-count-satisifed is also an activation condition for all sets. (Activation on every character can be achieved by requesting one-character read operations.)

The desired activation set is requested with the following call:

CAL1,8 fpt

where fpt points to the FPT shown below.



where

n = 0 for normal activation set.

n = 1 for the special graphics and control characters defined below.

n = 2 for the teletype control characters defined below.

n = 3 for EOT activation on 2741s.

The special graphics characters are

] [} { \ " = ' @ # : ? > _ % , ^ / - ¬ ;) * $! &
| + (< . ¢

The control characters are

SOH, STX, ETX, HT, ACK, BEL, BS, ENQ, NAK, VT, SO, SI, DLE, DC2, DC4, SYN, ETB, CAN

All characters are transmitted to a reading program in their XDS Standard EBCDIC value (see Appendix A). Note that the control characters EM, ESC NUL (ignore), and DEL (RUBOUT) are not included in any set.

ON-LINE AND BATCH DIFFERENCES

The UTS Monitor responds differently to certain procedures depending on whether an on-line or a batch program issued the call. These differences are outlined below. (The procedures are discussed in the UTS/BP Reference Manual, 90 17 64.)

EXIT RETURN (M:EXIT)

Batch: The Monitor performs any PMDI dumps that have been specified for the program. It then reads the C device, ignoring everything up to the next control card.

On-line: The Monitor returns control to the on-line executive program (TEL) and, after sending a message, sends a prompt (!) character to the terminal. It then awaits additional commands.

ERROR RETURN (M:ERR)

Batch: The Monitor lists the message

!! JOB id ERRORED BY USER AT xxxx

where xxxx is the address of the last instruction executed in the program. The message plus the contents of the current register block and program status doubleword (PSD) are listed on the LL and DO devices. Postmortem dumps are performed and the C device is read; everything up to the next control command is ignored.

On-line: The Monitor lists the message

A800 YOU ISSUED AN ERROR OR ABORT CAL

The Monitor then returns control to the on-line executive (TEL), which sends a prompt character (!) to the terminal and awaits commands.

ABORT RETURN (M:xxx)

Batch: The Monitor lists the message

!! JOB id ABORTED BY USER AT xxxx

where xxxx is the address of the last instruction executed. This message plus the contents of the current register block and program status doubleword (PSD) are listed on the LL and DO device.

When a job is aborted, all specified postmortem dumps are performed but no further control commands are honored until a JOB or FIN control command is encountered.

On-line: The Monitor lists the message

A800 YOU ISSUED AN ERROR OR ABORT CALL

This message is listed on the UC device. The Monitor then returns control to the on-line executive, which sends a prompt character (!) to the terminal and awaits additional commands.

TYPE A MESSAGE (M:TYPE)

Batch: The Monitor lists the specified message on the OC device.

On-line: The Monitor lists the specified message on the UC device.

A variant of M:TYPE is M:MESSAGE which unconditionally lists a message on the operator's console (OC device). The format of M:MESSAGE is identical to that of M:TYPE except for the FPT code which is zero.

REQUEST A KEY-IN (M:KEYIN)

Batch: The Monitor lists the specified message on the OC device and enables the operator's reply to be returned to the user program. The ECB flag is set to zero when the reply is completed.

On-line: The Monitor lists the specified message on the UC device and enables the user's reply to be returned to the user program. A prompt character is sent to the terminal if it was specified by an M:PC. The ECB flag is set to zero when the reply is completed.

COMMENT TO INTERRUPT OR BREAK KEY (M:INT)

Batch: The purpose of this procedure is to set the address of a routine to be entered when the INTERRUPT button is depressed at the operator's console. When control is given to the INT routine as a result of an interrupt, the Monitor pushes the PSD and general registers into a 19-word block of user's memory (the user's TCB) on a doubleword boundary and places a pointer to word 0 of the PSD in register 1. The TRTN routine may be used to restore control to the user program.

On-line: The purpose of this procedure is to set the address of a routine to be entered when an interrupt is generated at an on-line terminal. When the BREAK key is depressed, the Monitor pushes the PSD and general registers into a 19-word block of user's memory (the user's TCB) on a doubleword boundary and places a pointer to word 0 of the PSD in register 1. The TRTN routine may be used to restore control to the user program.

10. COMMUNICATIONS SERVICES TO USER PROGRAMS

INTRODUCTION

Communication services are the functions performed by character-oriented communication (COC) routines for user programs. COC routines control the operation of input/output terminals, such as Teletype and 2741 terminals, that communicate with the computer a character at a time. The functions performed by COC routines include

1. Device handling for XDS Model 7611 Character-Oriented Communication hardware.

2. Character translation (unless suppressed) to and from internal EBCDIC codes and the external codes of the various types of terminals that may be attached to UTS. Terminal types include: Teletype Models 33, 35, and 37; XDS Model 7015 Teletypewriters; IBM 2741 Selectric and EBCD; and others by extension.

3. Parity generation and detection by character for those terminals requiring it.

4. Division of input character strings into messages as defined by receipt of activation characters (usually carriage return, line feed, form feed, and count complete, but other sets are specially available).

5. Communication with the UTS scheduler on break, read, read complete, output blocked, output unblocked, and other events that effect swap and execution scheduling.

6. Special interpretation of certain characters for intra-line editing and software control functions.

Input and output from COC terminals is stored in four-word blocks, each containing 14 characters plus a halfword link to the next related block. After a read operation is complete, the input message is moved from these buffers directly to the user's buffer area (BUF in M:READ). The actual number of characters received is reported in ARS (actual record size) of the DCB. On a write operation, the user output message (BUF in M:WRITE) is moved to COC buffers to await transmission. Unused COC buffers are held in an available pool. The user program is blocked appropriately when needed buffers are not available for output and restarted when they become available.

WRITE OPERATIONS

Records are written on a COC terminal using the M:WRITE procedure call. The WRITE routine moves the specified number of bytes from the user's buffer to a buffer in the COC routines. The write operation is always a "wait" operation. This means that control is returned to the user program after the character string has been transferred to the COC buffer but before it has been completely transmitted to the terminal. If record keys are specified, they are ignored.

Output in excess of 140 bytes from a single write CAL is ignored. If the specified record size is zero, no action is taken and no characters are transmitted. If more than three trailing blanks occur in an output record, all are suppressed.

If the output contains a NUL character (X'00') the write operation is terminated at that point; i.e., the zero byte and all remaining characters in the record are ignored.

Characters are transmitted to the terminal exactly as supplied, with the following exceptions. Certain characters such as FF and SUB are modified (see Table A-4). Whenever either a carriage return or line feed character is detected, the appropriate character pair (carriage return and line feed) is sent to the terminal to return the carrier.

If the write operation is through a DCB other than the M:UC DCB, say the M:LO or M:DO DCB, the COC routines automatically supply carriage return and line feed characters at the end of the character string unless a carriage return, SYNC, or line feed were the last characters in the buffer (see VFC in "Device and DCB calls" for special format control). This means that the number of bytes specified in the function parameter table is moved from the user's buffer area to COC buffers and the carriage return and line feed characters are appended in the COC buffers.

If the write is through the M:UC DCB, the carriage return and line feed characters are not automatically supplied. The user may therefore make up single lines through a series of writes (without carriage return characters) or may produce several lines at the terminal with a single write (by inserting several carriage return characters in the buffer).

For all write operations, a count of characters between carriage returns is maintained. This count is compared with the maximum for the physical terminal as specified with the PLATEN command. If the line is too long, additional carriage return and line feed characters are inserted to break the line unless the platen width is less than 12 characters. Line length is a parameter supplied at system generation time and is retained in the job information table (JIT). It may be altered with the TEL PLATEN command. A count of the lines on a page is also maintained and a page heading line is supplied to the terminal as outlined in the section "Page Control and Page Headings".

READ OPERATIONS

Records are read from a COC terminal using the M:READ procedure call. The READ routine causes the COC routines to accept input characters from the terminal. If a prompt character has been specified, it is sent to the terminal to signal that the COC routines are ready to accept input

characters. If characters have been typed ahead, they are echoed after the prompt is issued.

The read operation is always a "wait" operation. This means that the complete input message is transferred to the user's buffer area before control passes to the next instruction. Messages are completed on receipt of

1. The number of characters requested.
2. A carriage return character.
3. A line feed character.
4. A form feed character.
5. The FS, RS, GS, and US codes (L^{cs}, M^{cs}, N^{cs}, and O^{cs} keys).
6. The EOT and SUB codes (D^{c} and Z^{c} keys).
7. The end-of-file convention, ESC F.

The activation character (any item in 2-6 above) is the last character in the buffer. Additional special activation or termination characters are supplied when Delta initiates a read operation. They are

tab

^

,

=

/

The actual number of characters in the message received, including the activation character, is returned in word 4 (ARS) of the DCB. No more characters than specified in the M:READ functional parameter table are transferred to the user's buffer area. Read requests for zero bytes yields an abnormal code of 1D.

The response of COC routines to receipt of various end-of-message characters from a terminal is as follows:

Characters	Response
Carriage return or line feed	The appropriate characters are sent to the terminal to ensure a carrier return. However, the actual character received is placed in the buffer.
Form feed	The code FF (EBCDIC 0C) is placed in the buffer, a carriage return and line feed character pair is sent to the terminal, followed by page heading output.
FS,RS,GS,US,EOT	The carrier is not moved. The character is placed in the buffer, and the message is terminated.
Break	An underscore (left arrow on TTYs) is sent to the terminal, the carrier is returned, the message is deleted, and the break entry of the program, if any, is taken.
ESC F	The end-of-file exit from the read CAL is taken. Any characters preceding the ESC F are delivered to the reading program and appended with a carriage return character.

Other characters may act as message terminators if special activation sets are requested; see Change Activation Characters, Chapter 9.

Characters received with parity errors for terminals in the parity checking mode are identified by the SUB code (EBCDIC 1A) which is placed in the buffer. For these characters, a number character, #, is returned to the terminal.

Bad information, such as a character parity error, is reported via the lost-data (07) code to the abnormal CAL exit, if it exists. If no abnormal exit is specified, then the bad information is not reported.

In addition to the line cancel, which may be initialized by the ESC X keys, individual characters may be deleted by the RUBOUT key. In this case, the last character typed is removed from the COC buffer and a backslash character (\) is sent to the terminal. A number of characters, n, may be deleted by typing the rubout character n times. If the first character of a line is deleted, the response is as if ESC X were received.

The user program or processor may set up a prompt character to be delivered to the terminal just prior to each read. The prompt character is set by using the M:PC procedure call described in Chapter 9. Any valid EBCDIC character may be specified. A null character (EBCDIC 00) turns off the prompt action.

Since the prompt character is carried in JIT for each user, the TEL and Delta processors do not prompt via this mechanism. They prompt by writing single character records before issuing a read.

ERROR AND ABNORMAL CONTROL

Error returns occur in the following cases:

1. Bad DCB address (CAL error return)
2. Bad buffer address (DCB error return)

CAL abnormal returns are taken for

1. Lost data (TYC=2) - parity errors in received message or insufficient COC buffers.

2. Beginning-of-tape (TYC=3) - CAL not read or write, bad line number, or zero byte count.

3. End-of-file (TYC=7) - ESC F character pair received.

If no error return is specified, control is returned to TEL and an error message is typed on the terminal.

BREAK CONTROL

Action on receipt of the break character depends on whether the terminal is reading or not. If reading, the carrier is returned and the message, if any, is deleted. The current read operation is terminated.

Whether reading or writing, control goes to an alternate address associated with the user program, and the user program status doubleword (PSD) and registers, as of the point of interrupt, are placed in the users task control block (TCB) temporary stack. The program may be continued from the point of interrupt by giving a trap return (M:TRTN or CAL1,9 5). The actual alternate address used depends on the user program and associated processors in the following order:

1. If the user has issued a M:INT CAL, the address specified by that CAL is used. A zero or invalid address resets break control.

2. If Delta is associated with the program, then control goes to Delta.

3. If neither 1 nor 2 apply, then control goes to TEL. A message is typed and TEL issues a request for commands from the terminal.

In all of the above cases, all current output is transferred to the terminal; none is lost. Because of the blocking action of the COC routines, this output is not usually longer than four seconds or four seconds plus one line.

Break signals are counted by the COC handler. This is done to provide fail-safe operation against program errors in the user break handling routine, to allow special subprocessor action on multiple break signals and to provide compatible operation with future communication equipment that does not have full-duplex lines. If four break signals are received from a terminal without intervening characters, control is given to TEL as if a Monitor escape (Y^C) character had been received.

MONITOR ESCAPE

A terminal may always be put in communication with TEL by input of the Y^C character. No current output is lost but the current input line is canceled (characters for a left-facing arrow, a carriage return, and a line feed are sent and the carrier is returned) if the terminal is in read status. If the user program is restarted (via the CONTINUE or GO command) from the point of escape and the terminal was previously reading, the read is reissued.

SET AND DEVICE DCB CALs

The M:SETDCB CAL may be used to set abnormal and error addresses in a DCB associated with a terminal. Error codes and other information communicated to the user program is as specified in Appendix B. If no error address is specified in the DCB, control is transferred to TEL and a message is sent to the terminal.

Only certain M:DEVICE CALs are acknowledged by the COC routines. These CALs are listed in Table 28. All other CALs that set parameters in a DCB associated with a COC terminal are ignored without comment. In general, any CAL may be used and will result in the specified modification to the DCB but only the parameters listed in Table 28 are used by COC routines.

PAGE CONTROL AND HEADINGS

COC routines count the lines transmitted to and from a terminal. Whenever a read or write operation is initiated, this line count is compared with the limit for the terminal. If the maximum has been exceeded, a new page heading is produced. (The maximum may be exceeded by several lines if several input lines have been canceled via the X^C keys at the bottom of the page before the next read or write call is issued. If this occurs an appropriate adjustment is made in the heading.)

Page headings are also produced whenever an M:DEVICE call specifying PAGE is issued by a user program or the characters "FF" (L^C) are entered into the terminal. This case is similar to page overflow in that heading information is not produced until the associated user program or processor issues its next read or write call.

Two kinds of page headings are produced:

1. The standard page heading.

2. A user heading as specified by HEADER and COUNT in a device call.

Heading information is taken from the DCB associated with the read or write call. Thus, if write calls are issued through several DCBs, the heading printed will depend on the DCB associated with the call that produced the page overflow.

The standard page heading includes current time, date, user identification and account number, user identification and line number, page number, and possibly an administrative message. The heading is typed on the top line of the form just under the fold (if any). The heading information is preceded by six blank lines (fewer if excess lines were printed

Table 28. M:DEVICE Parameters Acknowledged by COC Routines

Parameter Set by M:DEVICE CAL	COC Action
PAGE	Page heading is typed on the terminal (see "Page Control and Page Heading").
LINES	Number of printable lines per page is set.
NLINES	The current number of lines on the terminal page is contained in the JIT (byte JB:LC).
SIZE	Record size (in bytes) used by read and write CALs for which no size is specified. If record size is not specified in either the CAL FPT or the DCB, no characters are transmitted and return is immediate.
SPACE	Number of indicated spaces minus one are inserted before each write if VFC is not on and SPACE is set. Counts 0 and 1 result in single spacing (no spaces are inserted before each write).
VFC	COC routines simulate the printer's vertical format control as specified in the first character of the text line if VFC is set. The simulation is limited to the following cases: Hex Code — Action C1 - CF — COC routines insert 1-15 spaces before the print line. (Page check on each insert.) F1 — COC routines skip to top of page and print the heading information followed by the print line. 60, E0 — COC routines do not insert carriage return and line feed characters after print line. In all cases except the latter, the print line is followed by a carriage return and line feed characters and a check for page overflow.
DRC/NORDC	Used to inhibit automatic page heading if the mode is BCD. Used to control transparent mode if BIN is specified. (See "Transparent Mode" section.)
COUNT	See Page Control and Page Headings section.
HEADER	See Page Control and Page Headings section.
TABS	See TABS section.

on the preceding page). It is followed by five blank lines. With a standard of 54 printed lines to a page, this spacing produces 11-inch pages with one-inch margins at top and bottom. The standard heading line may be omitted, if desired, by setting DRC in the DCB or by setting the page length less than 11 lines.

Example:

12:01 12/12/69 ACCT NAME 1A-03[36] Administrative Message
1 2 3 4 5 6 7

1. Time the page heading was issued (24-hour clock).
2. Current date.
3. Log-on account.
4. Log-on identification.
5. Scheduler's job identification (ID) and line number of COC line.
6. Page number, enclosed in brackets, centered for a platen 72 characters wide.
7. Administrative message (limited to 64 characters) supplied to all terminals by system operator via this mechanism.

User headings, which are specified in the DCB of the read or write call, are provided following the automatic heading. The position, text, and page numbers of these headings are as specified in the UTS/BP Reference Manual, 90 17 64. The page count in this heading is that carried in the DCB and and is reset with each COUNT device call while page count for the standard heading is carried in the JIT and is never reset.

TAB SIMULATION

TAB stops that are set in output DCBs by a device call specifying TAB, by a SET command, or by the TEL command TABS, cause spaces to be sent to the terminal. These spaces bring the current position of the carrier to that indicated by the next higher tab stop in the DCB. The platen width test is still in effect and the carrier is returned if the count-on-line exceeds the platen width. If tab simulation is not in effect, the tab character is sent directly to the terminal. If tab simulation is on but no tab stops are set, one space is sent for each tab character.

Tabs received in the input stream are handled similarly, except that a tab is always echoed by at least one space (if echoplexing is on).

Three things are necessary for tab simulation to take effect:

1. Tab simulation must be on (ESC and T control).
2. Tab stops must be set in the M:UC DCB or the DCB controlling read or write.
3. Tab characters must be sent or received.

Simulation of tab stops is turned off and on by the user via the character pair ESC-T. These characters are not transmitted to the reading program and each pair switches tab simulation flag from on to off or vice versa. When the flag is on and a tab character (ANSCII 09) is received, enough blanks are sent to the terminal to move the carrier to the next higher tab position. When reading, the tab character is replaced by one or more spaces, as appropriate, in the input buffer if space-insertion mode is on; if off, the tab character is placed in the input buffer for the reading program. Space-insertion mode is toggled by the ESC-S character pair. Carriage returns are not inserted to split extra long input lines created this way.

When in effect, the tab stops used for simulation are obtained in the following order:

Output

1. If tab stops are set in the calling DCB, they are used.
2. If tab stops are set in M:UC DCB, they are used.
3. Tabs are replaced with a single space.

Input

1. If tab stops are set in the M:UC DCB, they are used.
2. A single space is echoed for each tab.

In all cases in which tabs are set but the current carrier position is beyond any tab stop that is set, the tab is replaced with a single space.

TRANSPARENT MODE

The transparent mode for input or output is controlled by setting the DRC and BIN mode flags in the DCB. If DRC and BIN are set, the transparent mode is in effect. This will cause all input and output through that DCB to be passed literally (i.e., no translation or interpretation will be done). The transparent mode may be escapted from by depressing BREAK. This mode of operation is not allowed for 2741 terminals.

APPENDIX A. XDS STANDARD SYMBOLS, CODES AND CORRESPONDENCES

XDS STANDARD SYMBOLS AND CODES

The symbols listed here include two types: graphic symbols and control characters. Graphic symbols are displayable and printable; control characters are not. Hybrids are SP (the symbol for a blank space), and DEL (the delete code) which is not considered a control command.

Two types of code are also shown: (1) the 8-bit XDS Standard Computer Code, i.e., the XDS Extended Binary-Coded-Interchange Code (EBCDIC); and (2) the 7-bit American National Standard Code for information Interchange (ANSCII), i.e., the XDS Standard Communication Code.

XDS STANDARD CHARACTER SETS

1. EBCDIC

   57-character set: uppercase letters, numerals, space, and & - / . < > ( ) + | $ * : ; , % # @ ' =

   63-character set: same as above plus ¢ ! _ ? " ¬

   89-character set: same as 63-character set plus lowercase letters

2. ANSCII

   64-character set: uppercase letters, numerals, space, and ! " $ % & ' ( ) * + , - . / \ ; : = < > ? @ _ [ ] ^ # | ¬

   95-character set: same as above plus lowercase letters and { } ¦ ~ `

CONTROL CODES

In addition to the standard character sets listed above, the XDS symbol repertoire includes 37 control codes and the hybrid code DEL (hybrid code SP is considered part of all character sets). These are listed in the table titled XDS Standard Symbol-Code Correspondences.

SPECIAL CODE PROPERITIES

The following two properties of all XDS standard codes will be retained for future standard code extensions:

1. All control codes, and only the control codes, have their two high-order bits equal to "00". DEL is not considered a control code.

2. No two graphic EBCDIC codes have their seven low-order bits equal.

Table A-1. XDS Standard 8-Bit Computer Codes (EBCDIC)

Least Significant Digits	Hexadecimal	Binary	Most Significant Digits 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
			0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0	0000	NUL	DLE	ds [6]		SP	&	-									0
1	1	0001	SOH	DC1	ss				/		a	j		\ [1]	A	J		1
2	2	0010	STX	DC2	fs						b	k	s	{ [1]	B	K	S	2
3	3	0011	ETX	DC3	si						c	l	t	} [1]	C	L	T	3
4	4	0100	EOT	DC4							d	m	u	[[1]	D	M	U	4
5	5	0101	HT	LF NL			Not normally assigned [7]				e	n	v	] [1]	E	N	V	5
6	6	0110	ACK	SYN [8]							f	o	w		F	O	W	6
7	7	0111	BEL	ETB							g	p	x		G	P	X	7
8	8	1000	EOM BS	CAN							h	q	y		H	Q	Y	8
9	9	1001	ENQ	EM	CR only						i	r	z		I	R	Z	9
A	10	1010	NAK	SUB	LF only		¢ [2]	!	¬ [1]	:								
B	11	1011	VT	ESC			.	$	,	#								
C	12	1100	FF	FS			<	*	%	@					Not normally assigned [7]			
D	13	1101	CR	GS			(	)	_	'								
E	14	1110	SO	RS			+	;	>	=								
F	15	1111	SI	US			| [2]	¬ [2]	?	"								DEL
					3		4				5							

NOTES:

1 The characters ^ \ { } [ ] are ANSCII characters that do not appear in any of the XDS EBCDIC-based character sets, though they are shown in the EBCDIC table.

2 The characters ¢ | ¬ appear in the XDS 63- and 89-character EBCDIC sets but not in either of the XDS ANSCII-based sets. However, XDS software translates the characters ¢ | ¬ into ANSCII characters as follows:

EBCDIC	=	ANSCII
¢		` (6-0)
|		¦ (7-12)
¬		~ (7-14)

3 The EBCDIC control codes in columns 0 and 1 and their binary representation are exactly the same as those in the ANSCII table, except for two interchanges: LF/NL with NAK, and HT with ENQ.

4 Characters enclosed in heavy lines are included only in the XDS standard 63- and 89-character EBCDIC sets.

5 These characters are included only in the XDS standard 89-character EBCDIC set.

6 Line feed has been assigned the EBCDIC value of X'20'. Line feed allows a user to continue to output on a new line without affecting the carrier position.

7 APL characters are assigned EBCDIC values that fall within the "not normally assigned" area of the standard XDS code set. These assignments are for APL internal use and are only reflected in 2741-APL translation tables.

8 Placing a SYN code as the last position of a nontransparent message will prevent the normal message appendage of the CR/LF pair. This allows a user to continue writing more than one message on the same line without affecting the carrier position. The EBCDIC SYN code is translated to an idle (IL) on output to 2741 terminals.

Table A-2. XDS Standard 7-Bit Communication Codes (ANSCII)

Least Significant Digits	Decimal (rows) (col's.) →	Binary [1]	Most Significant Digits 0	1	2	3	4	5	6	7
	↓		x000	x001	x010	x011	x100	x101	x110	x111
0	0	0000	NUL	DLE	SP	0	@	P	`	p
1	1	0001	SOH	DC1	! [5]	1	A	Q	a	q
2	2	0010	STX	DC2	"	2	B	R	b	r
3	3	0011	ETX	DC3	#	3	C	S	c	s
4	4	0100	EOT	DC4	$	4	D	T	d	t
5	5	0101	ENQ	NAK	%	5	E	U	e	u
6	6	0110	ACK	SYN	&	6	F	V	f	v
7	7	0111	BEL	ETB	'	7	G	W	g	w
8	8	1000	BS	CAN	(	8	H	X	h	x
9	9	1001	HT	EM	)	9	I	Y	i	y
10	10	1010	LF NL	SUB	*	:	J	Z	j	z
11	11	1011	VT	ESC	+	;	K	[[5]	k	{
12	12	1100	FF	FS	,	<	L	\	l	¦
13	13	1101	CR	GS	-	=	M	] [5]	m	}
14	14	1110	SO	RS	.	>	N	^ [4,5]	n	~ [4]
15	15	1111	SI	US	/	?	O	_ [4]	o	DEL
					2		3			

NOTES:

1 Most significant bit, added for 8-bit format, is either 0 or an even-parity bit for the remaining 7 bits.

2 Columns 0-1 are control codes.

3 Columns 2-5 correspond to the XDS 64-character ANSCII set.
Columns 2-7 correspond to the XDS 95-character ANSCII set.

4 On many current teletypes, the symbol

^	is	↑	(5-14)
_	is	←	(5-15)
~	is	ESC or ALTMODE control	(7-14)

and none of the symbols appearing in columns 6-7 are provided. Except for the three symbol differences noted above, therefore, such teletypes provide all the characters in the XDS 64-character ANSCII set. (The XDS 7015 Remote Keyboard Printer provides the 64-character ANSCII set also, but prints ^ as ∧. It also interprets the [ ] characters as | ¬.)

5 On the XDS 7670 Remote Batch Terminal, the symbol

!	is	|	(2-1)
[	is	¢	(5-11)
]	is	!	(5-13)
^	is	¬	(5-14)

and none of the symbols appearing in columns 6-7 are provided. Except for the four symbol differences noted above, therefore, this terminal provides all the characters in the XDS 64-character ANSCII set.

Table A-3. XDS Standard Symbol-Code Correspondences

EBCDIC[t] Hex.	EBCDIC[t] Dec.	Symbol	Card Code	ANSCII[tt]	Meaning	Remarks
00	0	NUL	12-0-9-8-1	0-0	null	00 through 23 and 2F are control codes.
01	1	SOH	12-9-1	0-1	start of header	
02	2	STX	12-9-2	0-2	start of text	
03	3	ETX	12-9-3	0-3	end of text	
04	4	EOT	12-9-4	0-4	end of transmission	
05	5	HT	12-9-5	0-9	horizontal tab	
06	6	ACK	12-9-6	0-6	acknowledge (positive)	
07	7	BEL	12-9-7	0-7	bell	
08	8	BS or EOM	12-9-8	0-8	backspace or end of message	EOM is used only on XDS Keyboard/ Printers Models 7012, 7020, 8091, and 8092.
09	9	ENQ	12-9-8-1	0-5	enquiry	
0A	10	NAK	12-9-8-2	1-5	negative acknowledge	
0B	11	VT	12-9-8-3	0-11	vertical tab	
0C	12	FF	12-9-8-4	0-12	form feed	
0D	13	CR	12-9-8-5	0-13	carriage return	
0E	14	SO	12-9-8-6	0-14	shift out	
0F	15	SI	12-9-8-7	0-15	shift in	
10	16	DLE	12-11-9-8-1	1-0	data link escape	
11	17	DC1	11-9-1	1-1	device control 1	
12	18	DC2	11-9-2	1-2	device control 2	
13	19	DC3	11-9-3	1-3	device control 3	
14	20	DC4	11-9-4	1-4	device control 4	
15	21	LF or NL	11-9-5	0-10	line feed or new line	
16	22	SYN	11-9-6	1-6	sync	Idle for 2741 terminals.
17	23	ETB	11-9-7	1-7	end of transmission block	
18	24	CAN	11-9-8	1-8	cancel	
19	25	EM	11-9-8-1	1-9	end of medium	
1A	26	SUB	11-9-8-2	1-10	substitute	Replaces characters with parity error.
1B	27	ESC	11-9-8-3	1-11	escape	
1C	28	FS	11-9-8-4	1-12	file separator	
1D	29	GS	11-9-8-5	1-13	group separator	
1E	30	RS	11-9-8-6	1-14	record separator	
1F	31	US	11-9-8-7	1-15	unit separator	
20	32	ds	11-0-9-8-1		digit selector	20 through 23 are used with Sigma 7 EDIT BYTE STRING (EBS) instruction – not input/output control codes. 24 through 2E are unassigned.
21	33	ss	0-9-1		significance start	
22	34	fs	0-9-2		field separation	
23	35	si	0-9-3		immediate significance start	
24	36		0-9-4			
25	37		0-9-5			
26	38		0-9-6			
27	39		0-9-7			
28	40		0-9-8			
29	41		0-9-8-1			
2A	42		0-9-8-2			
2B	43		0-9-8-3			
2C	44		0-9-8-4			
2D	45		0-9-8-5			
2E	46		0-9-8-6			
2F	47		0-9-8-7			
30	48		12-11-0-9-8-1			30 through 3F are unassigned.
31	49		9-1			
32	50		9-2			
33	51		9-3			
34	52		9-4			
35	53		9-5			
36	54		9-6			
37	55		9-7			
38	56		9-8			
39	57		9-8-1			
3A	58		9-8-2			
3B	59		9-8-3			
3C	60		9-8-4			
3D	61		9-8-5			
3E	62		9-8-6			
3F	63		9-8-7			

[t] Hexadecimal and decimal notation.

[tt] Decimal notation (column-row).

Table A-3. XDS Standard Symbol-Code Correspondences (cont.)

EBCDIC† Hex.	EBCDIC† Dec.	Symbol	Card Code	ANSCII††	Meaning	Remarks
40	64	SP	blank	2-0	blank	
41	65		12-0-9-1			41 through 49 will not be assigned.
42	66		12-0-9-2			
43	67		12-0-9-3			
44	68		12-0-9-4			
45	69		12-0-9-5			
46	70		12-0-9-6			
47	71		12-0-9-7			
48	72		12-0-9-8			
49	73		12-8-1			
4A	74	¢ or ‘	12-8-2	6-0	cent or accent grave	Accent grave used for left single quote. On model 7670, ‘ not available, and ¢ = ANSCII 5-11.
4B	75	.	12-8-3	2-14	period	
4C	76	<	12-8-4	3-12	less than	
4D	77	(	12-8-5	2-8	left parenthesis	
4E	78	+	12-8-6	2-11	plus	
4F	79	| or ¦	12-8-7	7-12	vertical bar or broken bar	On Model 7670, ¦ not available, and | = ANSCII 2-1.
50	80	&	12	2-6	ampersand	
51	81		12-11-9-1			51 through 59 will not be assigned.
52	82		12-11-9-2			
53	83		12-11-9-3			
54	84		12-11-9-4			
55	85		12-11-9-5			
56	86		12-11-9-6			
57	87		12-11-9-7			
58	88		12-11-9-8			
59	89		11-8-1			
5A	90	!	11-8-2	2-1	exclamation point	On Model 7670, ! is |.
5B	91	$	11-8-3	2-4	dollars	
5C	92	*	11-8-4	2-10	asterisk	
5D	93	)	11-8-5	2-9	right parenthesis	
5E	94	;	11-8-6	3-11	semicolon	
5F	95	~ or ¬	11-8-7	7-14	tilde or logical not	On Model 7670, ~ is not available, and ¬ = ANSCII 5-14.
60	96	-	11	2-13	minus, dash, hyphen	
61	97	/	0-1	2-15	slash	
62	98		11-0-9-2			62 through 69 will not be assigned.
63	99		11-0-9-3			
64	100		11-0-9-4			
65	101		11-0-9-5			
66	102		11-0-9-6			
67	103		11-0-9-7			
68	104		11-0-9-8			
69	105		0-8-1			
6A	106	^	12-11	5-14	circumflex	On Model 7670 ^ is ¬. On Model 7015 ^ is ∧ (caret).
6B	107	,	0-8-3	2-12	comma	
6C	108	%	0-8-4	2-5	percent	
6D	109	_	0-8-5	5-15	underline	Underline is sometimes called "break character"; may be printed along bottom of character line.
6E	110	>	0-8-6	3-14	greater than	
6F	111	?	0-8-7	3-15	question mark	
70	112		12-11-0			70 through 79 will not be assigned.
71	113		12-11-0-9-1			
72	114		12-11-0-9-2			
73	115		12-11-0-9-3			
74	116		12-11-0-9-4			
75	117		12-11-0-9-5			
76	118		12-11-0-9-6			
77	119		12-11-0-9-7			
78	120		12-11-0-9-8			
79	121		8-1			
7A	122	:	8-2	3-10	colon	
7B	123	#	8-3	2-3	number	
7C	124	@	8-4	4-0	at	
7D	125	'	8-5	2-7	apostrophe (right single quote)	
7E	126	=	8-6	3-13	equals	
7F	127	"	8-7	2-2	quotation mark	

†Hexadecimal and decimal notation.
††Decimal notation (column-row).

Table A-3. XDS Standard Symbol-Code Correspondences (cont.)

EBCDIC† Hex.	EBCDIC† Dec.	Symbol	Card Code	ANSCII††	Meaning	Remarks
80	128		12-0-8-1			80 is unassigned.
81	129	a	12-0-1	6-1		81-89, 91-99, A2-A9 comprise the lowercase alphabet. Available only in XDS standard 89- and 95-character sets.
82	130	b	12-0-2	6-2		
83	131	c	12-0-3	6-3		
84	132	d	12-0-4	6-4		
85	133	e	12-0-5	6-5		
86	134	f	12-0-6	6-6		
87	135	g	12-0-7	6-7		
88	136	h	12-0-8	6-8		
89	137	i	12-0-9	6-9		
8A	138		12-0-8-2			8A through 90 are unassigned.
8B	139		12-0-8-3			
8C	140		12-0-8-4			
8D	141		12-0-8-5			
8E	142		12-0-8-6			
8F	143		12-0-8-7			
90	144		12-11-8-1			
91	145	j	12-11-1	6-10		
92	146	k	12-11-2	6-11		
93	147	l	12-11-3	6-12		
94	148	m	12-11-4	6-13		
95	149	n	12-11-5	6-14		
96	150	o	12-11-6	6-15		
97	151	p	12-11-7	7-0		
98	152	q	12-11-8	7-1		
99	153	r	12-11-9	7-2		
9A	154		12-11-8-2			9A through A1 are unassigned.
9B	155		12-11-8-3			
9C	156		12-11-8-4			
9D	157		12-11-8-5			
9E	158		12-11-8-6			
9F	159		12-11-8-7			
A0	160		11-0-8-1			
A1	161		11-0-1			
A2	162	s	11-0-2	7-3		
A3	163	t	11-0-3	7-4		
A4	164	u	11-0-4	7-5		
A5	165	v	11-0-5	7-6		
A6	166	w	11-0-6	7-7		
A7	167	x	11-0-7	7-8		
A8	168	y	11-0-8	7-9		
A9	169	z	11-0-9	7-10		
AA	170		11-0-8-2			AA through B0 are unassigned.
AB	171		11-0-8-3			
AC	172		11-0-8-4			
AD	173		11-0-8-5			
AE	174		11-0-8-6			
AF	175		11-0-8-7			
B0	176		12-11-0-8-1			
B1	177	\	12-11-0-1	5-12	backslash	
B2	178	{	12-11-0-2	7-11	left brace	
B3	179	}	12-11-0-3	7-13	right brace	
B4	180	[	12-11-0-4	5-11	left bracket	On Model 7670, [is ¢.
B5	181	]	12-11-0-5	5-13	right bracket	On Model 7670,] is !.
B6	182		12-11-0-6			B6 through BF are unassigned.
B7	183		12-11-0-7			
B8	184		12-11-0-8			
B9	185		12-11-0-9			
BA	186		12-11-0-8-2			
BB	187		12-11-0-8-3			
BC	188		12-11-0-8-4			
BD	189		12-11-0-8-5			
BE	190		12-11-0-8-6			
BF	191		12-11-0-8-7			

†Hexadecimal and decimal notation.

††Decimal notation (column-row).

Table A-3. XDS Standard Symbol-Code Correspondences (cont.)

EBCDIC† Hex.	EBCDIC† Dec.	Symbol	Card Code	ANSCII††	Meaning	Remarks
C0	192		12-0			C0 is unassigned.
C1	193	A	12-1	4-1		C1-C9, D1-D9, E2-E9 comprise the uppercase alphabet.
C2	194	B	12-2	4-2		
C3	195	C	12-3	4-3		
C4	196	D	12-4	4-4		
C5	197	E	12-5	4-5		
C6	198	F	12-6	4-6		
C7	199	G	12-7	4-7		
C8	200	H	12-8	4-8		
C9	201	I	12-9	4-9		
CA	202		12-0-9-8-2			CA through CF will not be assigned.
CB	203		12-0-9-8-3			
CC	204		12-0-9-8-4			
CD	205		12-0-9-8-5			
CE	206		12-0-9-8-6			
CF	207		12-0-9-8-7			
D0	208		11-0			D0 is unassigned.
D1	209	J	11-1	4-10		
D2	210	K	11-2	4-11		
D3	211	L	11-3	4-12		
D4	212	M	11-4	4-13		
D5	213	N	11-5	4-14		
D6	214	O	11-6	4-15		
D7	215	P	11-7	5-0		
D8	216	Q	11-8	5-1		
D9	217	R	11-9	5-2		
DA	218		12-11-9-8-2			DA through DF will not be assigned.
DB	219		12-11-9-8-3			
DC	220		12-11-9-8-4			
DD	221		12-11-9-8-5			
DE	222		12-11-9-8-6			
DF	223		12-11-9-8-7			
E0	224		0-8-2			E0, E1 are unassigned.
E1	225		11-0-9-1			
E2	226	S	0-2	5-3		
E3	227	T	0-3	5-4		
E4	228	U	0-4	5-5		
E5	229	V	0-5	5-6		
E6	230	W	0-6	5-7		
E7	231	X	0-7	5-8		
E8	232	Y	0-8	5-9		
E9	233	Z	0-9	5-10		
EA	234		11-0-9-8-2			EA through EF will not be assigned.
EB	235		11-0-9-8-3			
EC	236		11-0-9-8-4			
ED	237		11-0-9-8-5			
EE	238		11-0-9-8-6			
EF	239		11-0-9-8-7			
F0	240	0	0	3-0		
F1	241	1	1	3-1		
F2	242	2	2	3-2		
F3	243	3	3	3-3		
F4	244	4	4	3-4		
F5	245	5	5	3-5		
F6	246	6	6	3-6		
F7	247	7	7	3-7		
F8	248	8	8	3-8		
F9	249	9	9	3-9		
FA	250		12-11-0-9-8-2			FA through FE will not be assigned.
FB	251		12-11-0-9-8-3			
FC	252		12-11-0-9-8-4			
FD	253		12-11-0-9-8-5			
FE	254		12-11-0-9-8-6			
FF	255	DEL	12-11-0-9-8-7		delete	Special – neither graphic nor control symbol.

†Hexadecimal and decimal notation.

††Decimal notation (column-row).

Table A-4. ANSCII Control-Character Translation Table

Input					Output	
ANSCII	TTY Key	Echoed	Prog. Receives (EBCDIC)	Process	EBCDIC	Transmitted (ANSCII)
NUL (00)	P[CS]	None	None	None	NUL (00)	Nothing (end of output message).
SOH (01)†	A[C]	SOH	SOH	None	SOH (01)	SOH
STX (02)†	B[C]	STX	STX	None	STX (02)	STX
ETX (03)†	C[C]	ETX	ETX	None	ETX (03)	ETX
EOT (04)†	D[C]	EOT	EOT	None	EOT (04)	EOT
ENQ (05)†	E[C]	ENQ	ENQ (09)	None	HT (05)	Space(s) if tab simulation on, or HT (09) if not.
ACK (06)†	F[C]	ACK	ACK	None	ACK (06)	ACK
BEL (07)	G[C]	BEL	BEL	None	BEL (07)	BEL
BS (08)	H[C]	BS	BS	None	BS (08)	BS
HT (09)	I[C]	Space to tab stop if tab simulation on, or 1 space if not.	Spaces to tab stop, or one space, or tab (05) depending on mode.	None	ENQ (09)	ENQ (05)
LF/NL (0A)	NL	CR and LF	LF (15)	Input Complete	NAK (0A)	NAK (15)
VT (0B)	K[C]	VT	VT	None	VT (0B)	VT
FF (0C)	L[C]	None	FF	Page Header	FF (0C)	Page Header
CR (0D)	CR	CR and LF	CR	Input Complete	CR (0D)	CR and LF (0A)
SO (0E)	N[C]	SO	SO	None	SO (0E)	SO
SI (0F)	O[C]	SI	SI	None	SI (0F)	SI
DLE (10)†	P[C]	DLE	DLE	None	DLE (10)	DLE
DC1 (11)	Q[C]	DC1	DC1	None	DC1 (11)	DC1
DC2 (12)	R[C]	DC2	DC2	None	DC2 (12)	DC2
DC3 (13)	S[C]	DC3	DC3	None	DC3 (13)	DC3
DC4 (14)†	T[C]	DC4	DC4	None	DC4 (14)	DC4
NAK (15)†	U[C]	NAK	NAK (15)	None	LF/NL (15)	CR and LF (0A)
SYN (16)†	V[C]	SYN	SYN	None	SYN (16)	SYN
ETB (17)†	W[C]	ETB	ETB	None	ETB (17)	ETB
CAN (18)	X[C]	Back-arrow and CR/LF	None	Cancel input or output message.	CAN (18)	CAN
EM (19)	Y[C]	Back-arrow and CR/LF	None	Monitor Escape/ Control to TEL	EM (19)	EM
SUB (1A)	Z[C]	SUB	SUB	None	SUB (1A)	# (A3)
ESC (1B)	K[CS] ESC PREFIX	#(A3)	None	Initiate escape sequence mode.	ESC (1B)	ESC
FS (1C)	L[CS]	FS	FS	Input Complete	FS (1C)	FS
GS (1D)	M[CS]	GS	GS	Input Complete	GS (1D)	GS

Table A-4. ANSCII Control-Character Translation Table (cont.)

Input					Output	
ANSCII	TTY Key	Echoed	Prog. Receives (EBCDIC)	Process	EBCDIC	Transmitted (ANSCII)
RS (1E)	N^{cs}	RS	RS	Input Complete	RS (1E)	RS
US (1F)	O^{cs}	US	US	Input Complete	US (1F)	US
} (7D)	ALT-MODE	} or None	} or None	} if model 37; as ESC if model 33 or 35.	} (B3)	} (7E)
~ (7E)	ESC (7015)	~ or None	~ or None	~ if model 37; as ESC if model 35 or 7015.	¬ (5F)	~ (7E)
DEL (7F)	Rubout	\	None	Rubout last character.	DEL (FF)	None
All ANSCII upper and lower case alphabetics are translated on input into the corresponding EBCDIC graphics as shown in Tables A-1 and A-2. All special graphics map as shown, allowing for Table A-1, Note 2, and the exceptions above for model 33 and 35. Lower case alphabetics map into corresponding EBCDIC upper case if the ESC-U mode is set. Upper case alphabetics map into corresponding EBCDIC lower case if ESC-) is set.					Alphabetic and symbol output translation is also as shown in Tables A-1 and A-2; for Models 33 and 35, and 7015 terminals, however, lower case alphabetics are automatically translated to upper case.	

†These characters are communication control characters reserved for use by hardware. Any other use of them risks incompatibility with future hardware developments and is done so by the user at his own risk.

Table A-5. Substitutions for Nonexistent Characters on 2741 Keyboards

EBCDIC Character	APL Keyboard	Selectric® Keyboard	EBCD Keyboard
>	>	& on output only	>
<	<	# on output only	<
^	↑	¢	¢
|	|	° (degree)	|
¬	~	±	¬
#	≠	#	#
%	ρ	%	%
¢	⊂	¢	¢
@	α	@	@
"	∇	"	"
!	∘	!	!
&	∩	&	&
$	∪	$	$

APPENDIX B. MONITOR ERROR MESSAGES

INTRODUCTION

All Monitor error conditions are identified by an error code or by a unique user-defined error message. These conditions are reported to TEL by various parts of the Monitor and, when detected, cause TEL to examine a special ERRMSG file. If TEL does not find a message in the file for the error condition, it returns the error code and subcode (Tables B1-B5) to the terminal. If it finds a message in the file for the error condition, it sends the message to the terminal in place of the error code.

Two groups of Monitor error codes are defined in this section. They are I/O error and abnormal codes (Tables B1-B4) and other Monitor codes (Table B5). In both cases, a message is printed only if the Monitor has control. If the user asks for control, the error codes are returned to him. Otherwise, the Monitor takes unilateral action and prints the message corresponding to the code or the code itself if no message is in the ERRMSG file. Users who have taken control may return it for Monitor disposition by using M:MERC.

The error and abnormal addresses specified in a function parameter table (FPT) for a Read, Check, or Write function are temporary and are not retained by the Monitor between calls. Those addresses specified in an FPT for an Open function are retained in the specified data control block (DCB).

I/O error an abnormal conditions fall into two general categories:

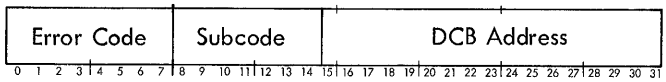
1. Those associated with insufficient or conflicting information.
2. Those associated with device failures or end-of-data conditions.

The Monitor responds to conditions of the first category by honoring the error and abnormal addresses in the associated DCB. The Monitor responds to conditions of the second category by honoring the error and abnormal addresses in the FPT for the associated Read, Check, or Write functions.

All Monitor error codes are in hexadecimal. The error and abnormal codes for conditions of the first category above are: 01, 02, 03, 08, 09, 0A, 13, 14, 15, 16, 17, 18, 19, 20, 2E, 40, 42, 43, 44, 46, 47, 4A, 4D, 51, 54, 55, and 56. Those for conditions of the second category above are: 04, 05, 06, 07, 1C, 1D, 41, 45, 49, and 57.

The Monitor communicates the error or abnormal code and the DCB address in SR3, and the location following the associated CAL1 in SR1. The code is contained in byte 0 of the word in SR3, a subcode is contained in bits 8-14, and the DCB address is contained in the rightmost 17 bits.

SR3



Note that the subcode field contains seven bits and an error code of 75/13 would appear as X'7526' in bits 0-15. (The first digit of the subcode is contained in bit positions 8, 9, and 10. Hence, it may have a value of 0-7.) The previous contents of SR1 and SR3 are lost. The meaning of each error and abnormal code is shown in Tables B-1 to B-4.

Table B-1. Abnormal Codes – Insufficient or Conflicting Information

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
01		OPEN	An attempt was made to open a DCB with insufficient information.
02		OPEN	An attempt was made to open the next file but there are no more files.
03		OPEN	The input or update file does not exist.
08		OPEN	An attempt was made to open the next file but the name of the next file is a synonym for the primary name of the file.
09	00	OPEN	The user privilege level was not high enough to allow issuing a direct device OPEN.
	01	OPEN	The device I/O address was not that of a symbiont device (card punch, card reader, line printer, or paper tape unit).
	02	OPEN	The device was already in use by another diagnostic program.
	03	OPEN	The device was currently in use by a symbiont. The operator must be asked to suspend the symbiont. The program should wait for this action before re-issuing the call.

Table B-1. Abnormal Codes – Insufficient or Conflicting Information (cont.)

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
0A		CLOSE	An attempt was made to close a DCB that is already closed.
13		DELREC or WRITE	The specified key was not found for an update file and the option is not NEWKEY.
14	00	OPEN	Any of the following may have occurred: (1) the Monitor has not received all information required to access the file, (2) access permission (read/write account numbers or password) has been violated for an existing file, (3) the file name length has been greater than 31 or less than 1, or (4) open NXTF was specified without room for synonymous file name in the DCB.
	01	OPEN	An attempt was made to open a file for output and another user or DCB has the file open for input or output.
15		DELREC or WRITE	An improper sequence of operations has been requested for an update file, or the FPARM address did not belong to the user. For example, a WRITE or DELREC was issued for a keyed file and there is no key given on the WRITE or DELREC.
16		WRITE	The NEWKEY option was specified, but the key already exists.
17		WRITE	The NEWKEY option was not specified for an output or scratch file.
18		WRITE	An attempt was made to write a keyed file sequentially with an out-of-order key.
19		OPEN/CLOSE	An illegal operation was attempted on M:UC DCB.
2E		OPEN	An attempt was made to open a DCB that is already open.

Note: In all of the above cases, return is made to the user's program for continuation of execution if no abnormal address is specified in the DCB.

Table B-2. Abnormal Codes – Device Failure or End-of-Data

Error Code	Originating Monitor Routine	Meaning of Code
04	PRECORD or READ	The beginning-of-file has been encountered.
05	PRECORD or READ	The end-of-data has been encountered.
06	READ	The end-of-file has been encountered (or first read of ! card).
07	READ	Data has been lost because the buffer was smaller than the record read, or a parity error was detected.
0B	OPEN	A read error has been encountered during labeled-tape sentinel processing, resulting in an unrecognized tape sentinel.
1C	READ, WRITE or PRECORD	The end-of-tape has been encountered.
1D	READ or PRECORD	The beginning-of-tape has been encountered or a bad command has been sent to the terminal.

Note: In all of the above cases, return is made to the user's program for continued execution if no abnormal address is specified in the I/O CAL FPT.

Table B-3. Error Codes – Insufficient or Conflicting Information

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
40		READ	A request was made to read an output file.
42		READ, WRITE, or RANDOM	The key was not valid. The key length was zero or greater than the key maximum for the file or a random file granule number is out of legal range.
43		READ	No record having the specified key was found.
44		WRITE	A request was made to write in an input file.
46	XX	READ	The DCB contains insufficient information to open a closed DCB on a Read operation. Subcodes corresponding to the OPEN error codes above describe why the error code did not take.
47	XX	WRITE	The DCB contains insufficient information to open a closed DCB on a Write operation. Subcodes corresponding to the OPEN error codes above describe why the error code did not take.
4A		READ or WRITE	Either the specified buffer or the indirect address in FPT does not belong to user.
51	XX	CLOSE	An attempt was made to close and save an output file that is open in IN mode through some other DCB.
54		READ	The user has tried to read a control command via the control input (C) device more than once through the same DCB.
55		OPEN	Too many files are open simultaneously (the Monitor's file-use tables cannot handle that many files).
56		CLOSE or CVOL	This RAD is saturated, or the system is unable to switch to the next tape volume because the reel number has not been specified.
75	01	READ	Data records were lost due to a bad RAD address in master index.
75	02	READ	The master index is inaccessible due to bad RAD address in preceding master index.
75	03	OPEN	The entire file is inaccessible due to bad RAD address in file directory or file information table.
75	04	OPEN	The file directory (and all files therein) is inaccessible due to a bad RAD address in file directory.
75	05	OPEN	All files in account were lost due to bad RAD address in account directory.
75	06	OPEN	A bad RAD address link to next account directory exists. The current account and other accounts are gone.

Note: In all of the above cases, the job is aborted if no error address is specified in the DCB. In batch mode, the Monitor skips to the next job; in on-line mode, control is returned to TEL which prints the message and awaits further user commands. For error code 54, the job is aborted in all cases.

Table B-4. Error Codes – Device Failure or End-of-Data

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
41	00	READ	An irrecoverable read error has occurred.
41	01	COOP	A bad RAD address was detected by the input cooperative when reading the input symbiont file.
45		WRITE	An irrecoverable write error has occurred.
49		WRITE	No tape is available, or user privilege level is insufficient.
57		READ or WRITE	The RAD is saturated, or the system is unable to switch to next tape volume because the reel number has not been specified.
Note: In all of the above cases, the job is aborted if no error address is specified in the I/O CAL FPT. In batch mode, the Monitor skips to the next job; in the on-line mode, control is returned to TEL which prints the message and awaits further user commands.			

Table B-5. Other Monitor Error Codes

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
A0		ASP	An attempt was made to RUN under an invalid debugger name, or a request for an invalid debugger through TEL.
A1		ASP	An attempt was made to associate a debugger with a shared processor.
A2			Unused.
A3		TRAP	Trap control cannot be given to the user because his task control block (TCB) does not exist or is full, or his pointer has been destroyed.
A4	00	TRAP	A user trap occurred.
	01		Trap 40 – Nonexistent instruction
	02		Trap 40 – Nonexistent memory reference
	03		Trap 40 – Privileged instruction
	04		Trap 40 – Memory protect violation
	05		Trap 41 – Unimplemented instruction
	06		Trap 42 – Stack overflow
	07		Trap 43 – Fixed point overflow
	08		Trap 44 – Floating point fault
	09		Trap 45 – Decimal arithmetic fault
	0A		Trap 46 – Watchdog timer
	0B		Trap 47 – Storage
	0C		Trap 4C – Parity error
A5		STEP	User's load module exceeds virtual core size limit.
A6			The Monitor cannot find the requested load module.
	30		Bad DCB or DCB table (B00).
A7		TEL	A program in progress was erased to make room for the latest user request.
A8		STEP	An error or abort CAL was issued. (RNST bits are also set.)
A9			Unused.
AA		STEP	A request was made for core library that does not exist.

Table B-5. Other Monitor Error Codes (cont.)

Error Code	Sub-code	Originating Monitor Routine	Meaning of Code
AB		OPEN	An invalid operational label was found in the DCB.
AC			An attempt was made to read the card reader by an on-line user.
AD			Unused.
AE		CALPROC ACTCP	The user issued a CAL with unknown codes.
AF		CALPROC	A CAL1 instruction references a non-DCB.
B0	00	DUMP	The program specified snapshot dumps but did not have an M:DO DCB.
	01		The program attempted snapshot dump of inaccessible or nonexistent memory.
	02		Inaccessible flag address given on conditional debug command.
B1		SEGLOAD	The Monitor cannot find the segment named in the user M:SEGLD.
B2			The user issued a CAL2, CAL3, or CAL4.
B3	00		Limit exceeded
	01		Punch
	02		Pages by processors
	03		Pages by user
	04		Pages through M:DO
	05		Permanent RAD storage
	06		Temporary RAD storage
	07		Scratch tapes
	08		Execution time
	09		RAD allocation
B4	00		Exit
	01		User issued M:ERR
	02		User issued M:XXX
	03		Operator E (error) key-in
	04		Operator X (abort) key-in
B5			Load and link (M:LINK) and load and transfer control (M:LDTRC) error messages:
	01		The user file cannot be opened.
	03		The user file does not exist.
	14		The user is denied access to his file.
	46		The file cannot be opened because of insufficient information.
	61		M:LINK and M:LDTRC are not permitted under Delta.
	62		M:LINK and M:LDTRC are not permitted when a shared processor is associated with the user program.
	63		The program must not be loaded with Link.
	64		The user must own all memory from data through dynamic data.
	65		The DCB is not in the DCB area.
	66		The user cannot get a blocking buffer.
	67		A logically impossible exit to Load and Link has occurred.
B6			M:LINK: Not SEGLOAD DCB.
B7			Too many buffers requested on POOL card.

APPENDIX C. COMPARISON OF UTS AND BTM TIME-SHARING SERVICES

INTRODUCTION

The following is a comparison of the time-sharing services of UTS to those of BTM from the terminal user's point of view. It is assumed that the reader is familiar with the BTM on-line capabilities described in the BTM/Reference Manual, 90 15 77.

TELETYPE OPERATIONS

Before dialing UTS for the first time, the terminal user accustomed to BTM time-sharing services should be aware of the fact that in UTS all on-line commands (except for DELTA) are terminated by a RETURN or LINE FEED. There is no system activation on two-characters of a name or on punctuation as in BTM.

The UTS terminal is activated with the same procedure used for BTM. Once the terminal is operational under UTS, the system responds by typing

```
UTS AT YOUR SERVICE
ON AT (date and time)
LOGON PLEASE:
```

The user inputs his account, name, and an optional password (in that order) as he would for BTM. There will be a short delay before UTS responds; the LOGON data must be printed on the operator's console first.

If the LOGON sequence is correct, the UTS response is quite elaborate compared to BTM: several lines on the terminal are skipped, a line or two of information is printed, several more lines are skipped, and finally, a prompt character (!) is printed. This response is due to the pagination feature of UTS; that is, the treatment of the terminal paper as if it were segmented into 8 1/2 by 11 inch pages with one-inch margins at the top and bottom of each page. Unless altered by a Terminal Executive Language (TEL) command, UTS assumes each page to be 54 lines long with 72 characters per line. Each page begins with a header. The header consists of the date, time, user's name, terminal id, page number, and operator's messages.

To terminate an on-line session, the user types the OFF command. This serves the same function as the BTM BYE command. UTS responds by typing the following statistics:

```
CPU=m.mmm    CON=n:mm    INT=nn    CHG=xxxx
```

where

CPU = the CPU time in minutes.

CON = the terminal time in hours and minutes.

INT = the number of terminal interactions during the session.

CHG = the number of charge units for the session.

Unlike BTM, the number of RAD and disk granules used during the on-line session are not printed.

If the user wants to log on again while the line is still connected to UTS, he does not have to hit the BREAK key as he would for BTM. All he must do is wait a few seconds and UTS will type the LOGON request again.

Several special teletype characters for UTS have different - or new - meanings from their BTM counterparts. These characters are listed in Table C-1.

INPUT/OUTPUT CONVENTIONS

The special CAL3's for terminal I/O in BTM are not implemented in UTS. Most of the services these calls provide, however, are available to the UTS on-line user in other forms:

CAL3,0	M:READ,M:KEYIN
CAL3,1	M:TYPE,M:WRITE,M:PRINT
CAL3,2	-
CAL3,3	-
CAL3,4	Batch: M:LINK,M:LDTRC
CAL3,5	On-Line: M:ASP,M:DSP
CAL3,6	M:EXIT (to TEL)
CAL3,7	-
CAL3,8	- (TCB available in JIT)
CAL3,9	-
CAL3,10	- (Error messages available in ERRFILE)
CAL3,11	-
CAL3,12	- (Mapped System does this automatically)
CAL3,13	M:JOB
CAL3,14	M:GL
CAL3,15	M:DATE,M:TIME

The fact reflects a fundamental aspect of the UTS system; that is, on-line users are treated in essentially the same way as batch users. In this case, it can be said that the great majority of system procedures available to one class of users are available to the other.

TERMINAL EXECUTIVE LANGUAGE (TEL) VERSUS BTM EXEC

Some of the new or different features provided by TEL as compared to BTM are

1. All TEL commands are terminated by a RETURN or LINE FEED. There is no system activation on two characters of a name or on punctuation as in BTM.

2. Many functions that had to be accomplished via a subsystem parameter in BTM can be accomplished by a single TEL command under UTS. Either such a function is carried out directly by TEL (e.g., submitting a batch

Table C-1. Special Teletype Characters for UTS

UTS Character	Response	BTM Character	Response	Meaning
RUBOUT or ESC RUBOUT	\	ESC RUBOUT	←	Erase last character
ESC RET or ESC LF	Local Lineation[tt]	ESC RET	Local Lineation[tt]	Local new line
ESC Y, ESC ESC, 4 BREAKS, or Y[ct]	!	ESC ESC, 4 BREAKS	!	Return to executive
ESC T	(none)	(none)		Toggle tab simulation
L[ct] or ESC L	Pagination[ttt]	(none)		End of page
ESC F	Lineation[tt]	(none)		End of file
ESC U	(none)	(none)		Toggle upper/lower case
ESC[ct]	(none)	(none)		Toggle tab relative mode
ESC S	(none)	(none)		Toggle space insertion mode
ESC (	(none)	(none)		Upper case shift
ESC)	(none)	(none)		Lower case shift
Q[ct] (X-on)	(none)	(none)		Turn on paper tape reader
S[ct] (X-off)	(none)	(none)		Turn off paper tape reader

[t]The superscript C indicates that the CONTROL key is to be depressed.

[tt]Lineation means that a carriage return and line feed are sent to the terminal.

[ttt]Pagination means that lines are skipped until the next terminal is reached and then a header is typed.

job) or an implicit call is made to the proper subsystem (e.g., the TEL command BUILD results in an implicit call to EDIT).

3. In contrast to the BTM subsystems FORTRAN, SYMBOL, and LOADER, the on-line user does not have to pre-assign his files to source input, binary output, and listing output for the corresponding UTS subsystems (FORT4, META, and LINK). In fact, all of the control commands needed to perform an assembly or load (assignment of DCBs, processor call, processor options) are combined into one TEL command.

4. UTS allows a properly-authorized user on-line access to peripheral devices (printer, punch, paper tape, card reader) and magnetic tape. Such capabilities do not exist for the BTM user.

5. Any load module under UTS may be called for execution by an on-line user via TEL. This includes load modules under any account (not just :SYS).

6. Whereas BTM recognizes the word HERE to mean the user's Teletype, UTS recognizes the word ME.

7. In UTS, a dollar sign may be used to refer to a program just assembled, compiled, or loaded during the current on-line session.

8. Under BTM, the only device-type assignment permitted is the assignment of a DCB to the user's terminal. With the UTS SET command, it is possible to set most of the DCB parameters which are set by the batch ASSIGN command and many of the parameters which are set by the BPM OPEN and DEVICE procedures.

9. TAB characters are handled somewhat differently in UTS. The effect of a tab character on input and output is dependent upon the DCB tab settings, the space insertion mode, the tab simulation mode, and the tab relative mode as described below:

On input of a tab character, the user's buffer receives:	Space Insertion Mode ON (Default)	Space Insertion Mode OFF
	N blanks according to the DCB tab settings and the effective carrier position. If the tab relative mode is in effect, the position of the carrier at the start of input is used to offset the DCB tab settings. This effectively compensates for prompt messages of varying lengths.	HT - horizontal tab character (X'05')

On input of a tab character, UTS will echo the following (unless echoplex is suppressed): and On output of a tab character, UTS will send the following:	TAB Simulation Mode ON (Default)	Tab Simulation Mode OFF
	N blanks according to the DCB tab settings and the carrier position	HT - horizontal tab character - to Teletype Models 35 and 37 and the 2741. A blank to the Teletype Model 33 and the 7015. (This will be sent even if echoplex is suppressed.)

A summary of TEL commands and the comparable BTM commands appear in Table C-2. The first column contains the TEL command format; the middle column contains the corresponding BTM command(s) required to achieve the same function. The command function is described in the third column. File identification is designated by "fid" and has the format:

UTS		BTM	
name	[.account .account.password ..password]	name	[(account) (account,password) (,password)]

The prompt character (!) has been left off the TEL and BTM EXEC commands. Prompt characters for subsystems, however, are indicated.

SUBSYSTEM COMPARISONS

In several cases, the BTM on-line subsystems are toned-down versions of more powerful processors available to batch users. In contrast, the UTS system allows both batch and on-line users access to many of the same processors. Differences between the UTS and BTM processors are described below:

UTS META AND BTM SYMBOL ASSEMBLERS

The on-line Meta-Symbol assembler for UTS, META, has several advantages over the BTM Symbol Assembler:

1. The limitations imposed by the Symbol language are lifted for the UTS on-line user. He can form as sophisticated assembly language programs as the UTS (and BPM) batch user.

2. The META Subsystem recognizes more assembly options than the BTM Symbol Subsystem:

AC (ac.,...,ac$_n$)	LO
BO*	LU
CI*	NS
CN	SD
CO	SI*
GO*	SO

*implicitly specified in the META command.

3. The parameters CI, SI, LO, BO, and GO do not need to be preassigned before calling META.

4. An on-line UTS user can update a CI file with a source file built under EDIT simply by specifying both files as input in the META command. (META can distinguish between the keyed records of the Edit file and the sequential binary records of the compressed file.)

UTS FORT4 AND BTM FORTRAN SUBSYSTEMS

The UTS FORT4 subsystem is an Extended FORTRAN IV compiler. The BTM FORTRAN subsystem is an Extended FORTRAN IV-H compiler. Since FORTRAN IV-H is a subset of FORTRAN IV, the UTS user can compile FORTRAN programs on-line which in BTM would have to be compiled in batch.

Note: When entering FORTRAN programs a line at a time, syntax checking is performed after each line is received. If a statement is to be continued, each comtinued line must end with a colon (:) and each continuation line must use column six. This is the exact opposite of BTM, where the colon indicates no continuation.

UTS LINK LOADER AND BTM LOADER SUBSYSTEMS

Both UTS LINK and BTM Loader Subsystems form nonoverlaid load modules from ROMs and libraries. Several options are available under LINK, however, which are not available under LOAD.

Take, for instance, internal symbol tables (ISTs). Considerable flexibility exists with regard to the construction of ISTs by LINK for use under DELTA. The user can specify whether

Table C-2. TEL Command Summary and Equivalent BTM Command(s)

TEL Command	BTM Command(s)	Description
BACKUP fid	(none)	Saves the specified file on a system tape.
BATCH fid	ASSIGN M:SI,(FILE,file) BPM INSERT JOB?Y	Enters the specified file in the batch job stream.
BUILD fid	EDIT *BUILD fid	Accepts a new file from the terminal.
COMMENT {ON / OVER} list where list is fid, LP, or ME	ASSIGN M:D0,(list) where list = FILE,name or HERE[1]	Directs error commentary (from an on-line assembler or compiler) to the specified device.
CONTINUE or GO	PROCEED	Continues processing from the point of termination.
COPYsf {TO / ON} df where sf = [DC/]fid df = [DC/]fid,LP, or ME	FERRET >C[OPY]fid_1,fid_2 or >E[XAMINE]fid	Copies a file to the specified device.
DELETE fid	FERRET >D[ELETE]fid	Deletes the specified file.
DISPLAY	FERRET >S[TATISTICS]	Lists the current values of various system parameters.
DONT COMMENT	(none)	Stops error commentary output.
DONT LIST	(none)	Stops listing output.
DONT OUTPUT	(none)	Stops object output.
EDIT fid	EDIT *EDIT fid	Calls Edit to modify a file.
FORT4[sp]...[,sp] [{ON / OVER}[rom][,list]] where sp=fid or ME rom=fid list=fid,LP, or ME	ASSIGN M:SI,(FILE,file) ASSIGN M:B0,(FILE,file) ASSIGN M:L0,(FILE,file) FORTRAN	Compiles the specified FORTRAN program.
GET fid	RESTORE fid	Restores the previously saved core image.

Table C-2. TEL Command Summary and Equivalent BTM Command(s) (cont.)

TEL Command	BTM Command(s)	Description
JOB jid	BPM STATUS CHECK?Y ID = jid	Requests the status of a previously entered batch job.
LINK[codes]rom[,rom][...,rom] [ON / OVER lmn] [;lid[,lid]... [,lid]] where rom = fid or $ lid = library fid codes include (L), (NL), (D), (ND), (C), (NC), (M), (NM)	LOAD ELEMENT FILES:[fid]... [fid] . . . SAVE lmn	Forms a load module as specified.
LIST{ON / OVER} list where list = fid, LP, or ME	ASSIGN M:LO,(list) where list = FILE,name or HERE	Directs the listing output to the specified device.
lmn[sp]...[,sp]... [ON / OVER [rom][,list]]	(none)	Initiates execution of a load module where sp is assigned to M:SI; rom is assigned to M:GO; list is assigned to M:LO.
MESSAGE text	FERRET >M[ESSAGE]text	Sends the specified message to the operator.
META[sp]...[,sp] [ON / OVER [rom][,list]] where sp = fid or ME rom = fid list = fid, LP, or ME	ASSIGN M:SI,(FILE,name) ASSIGN M:BO,(FILE,name) ASSIGN M:LO,(FILE,name) SYMBOL	Assembles the specified source program.
OFF	BYE	Disconnects terminal from system and provides accounting directory.
OUTPUT {ON / OVER} rom	ASSIGN M:BO,(FILE,name)	Directs rom output to a specified file.
PASSWORD xxxx	(none)	Assigns a new log-on password for the user.
PLATEN,l where l ≤ 11	MESSAGE OFF	Inhibits operator messages (No page header is printed for UTS).
PLATEN w[,l]	(none)	Sets the value of the terminal platen width and page length.

Table C-2. TEL Command Summary and Equivalent BTM Command(s) (cont.)

TEL Command	BTM Command(s)	Description
PRINT	(none)	Sends output to the line printer and card punch without waiting for the user to log off.
ESC ESC, Y^C, or 4 BREAKs followed by QUIT	Escape Command (ESC-ESC)	Terminates the current job step.
RUN[codes]rom[,rom]...[rom] [ON / OVER lmn][;lid[,lid]... [,lid]][UNDER DELTA / FDP] parameters same as in LINK	LOAD ELEMENT FILES: fid ... [fid] . . . XEQ ?Y	Loads the specified load module and starts execution (optionally under a debugging processor).
SAVE {ON / OVER} fid	SAVE fid	Saves the current core image on the designated file.
SET dcb 0	ASSIGN dcb	Clears DCB of previous parameters.
SET dcb [oplabel / device [tapeid] / tapecode] [;opt]...[;opt] where opt=device options	ASSIGN dcb (HERE)	Assigns device to a DCB or sets a DCB parameter.
SET dcb [tapecode[tapeid] / filecode[packid] /fid][;opt]...[;opt] where opt=file options	ASSIGN dcb (FILE,fid) [,(option) ...]	Assigns file to a DCB or sets a DCB parameter.
START[lmn / $] [UNDER DELTA]	RUN LOAD MODULE FID:lmn (executed under a subset of BTM DELTA)	Begins execution of a load module, either with or without an associated debugger.
STATUS	FERRET >S[TATISTICS]	Displays the current accounting values.
Subsystem Calls BASIC CONTROL DELTA EDIT FDP FORT4 LINK	 BASIC DELTA EDIT FORTRAN LOAD	These calls turn over control of the terminal executive to the subsystem.

Table C-2. TEL Command Summary and Equivalent BTM Command(s) (cont.)

TEL Command	BTM Command(s)	Description
Subsystem Calls (cont.) META PCL SUPER lmn (user's program)	 SYMBOL FERRET SUPER (none) BPM RUN	
TABS s[,s]...[,s] (maximum=16)	TABS [s]...[,s] (maximum=8)	Sets the simulated tab stops at the terminal.
TERMINAL type where type=33, 35, 37, or 7015	(none)	Sets the terminal type for proper I/O translations.

or not he wants an IST to be built for each input file. Also, the ISTs for several input files can be merged. These capabilities contrast to the BTM Loader Subsystem, whereby the D (debug) option allows only all-or-none IST construction.

In addition to the load map option (M) (available in both on-line loaders), LINK recognizes two other display options. The (D) option produces a list of all unsatisfied external and internal symbols at the completion of the linking process. The (C) option results in a display of all conflicting internal and external symbols. These displays may be inhibited by the (ND) and (NC) options. (BTM always outputs an undefined-external symbol map if debug is specified. It cannot output any undefined internal symbol map.)

Both on-line loaders search libraries to resolve unsatisfied external references. (Such a library is a file containing ROMs "linked" together.) LINK, however, does not restrict its search to the :BLIB file of any account, as does the BTM Loader Subsystem. Instead it searches any file specified in the library (lid) portion of the LINK command. In this way, the UTS user is relieved of maintaining all of his library-type ROMs in one unique file.

Link places code in the 00 and 01 protection type sections according to the dictates of the input ROMs. It does not force the entire load module into protection type 00, as does the BTM Loader.

UTS EDIT AND BTM EDIT

Features of UTS Edit that are different from those of BTM are listed below.

1. The following edit commands may be given via TEL:

   BUILD fid
   EDIT fid
   DELETE fid

2. The file identification (fid) must follow the UTS file identification structure.

3. UTS terminates the entire command if the BREAK key is depressed.

4. A new command is available, TA. It sets the tab positions and has the following format:

   TA {F / M / S}

   where

   F implies FORTRAN; tab set at column 7.

   M implies Meta-Symbol; tabs set at columns 10, 19, and 37.

   S implies Meta-Symbol, short form; tabs set at columns 8, 16, and 30.

   The actual tab simulation is carried out exactly like the TABS command.

UTS DELTA AND BTM DELTA

Features of UTS Delta that are different from those of BTM Delta are listed below:

1. Delta may be called by the following means:

   a. To load and execute a program under Delta, give the TEL command

      RUN rom UNDER DELTA

   b. To execute a load module under Delta, give the TEL command

      START lmn UNDER DELTA

   c. To call Delta after a program has already started executing, strike the CONTROL and Y keys simultaneously to return to TEL. Then give the TEL command DELTA.

d. To call Delta to write and check a short program, give the TEL command DELTA.

2. Symbol tables can be manipulated at load time (see Chapter 8).

3. The following new commands are available:

a. Symbol table control

;U	Display undefined symbols.
;KI	Remove current internal symbol tables.
;KG	Remove global symbol table and any symbols defined at terminal.

b. Execution control

)	Execute current instruction and display next one.

c. Memory searching and modification

e1,e2;W	Store e2 through mask in locations that match e1 through the mask.

d. Breakpoint control (data and transfer)

e,r,val,m;DR	Data breakpoint whenever contents of e, masked by m, are in relation r to val (r options are LS, EQ, GR, GQ, NQ, LQ).
e,r,val,m;DTr	Same as above in trace mode.
n;D	Remove nth data breakpoint.
0;D	Remove all data breakpoints.
;D	Display list of active data breakpoints.
;Y	Set transfer breakpoint mode.
;YT	Same as above in trace mode.
0;Y	Turn off transfer breakpoint mode.
loc;Y	Start transfer breakpoint execution mode at loc.
loc;YT	Same as above in trace mode.
l,m,n,o;YS	Set entries in SAT.
l,m,n,o;YR	Release entries in SAT.
;YR	Release all entries in SAT.
;YD	Display SAT.
loc,opt1,opt2;Y	Set transfer breakpoints as follows: opt1 = 0 all branches except those in SAT. opt1 = 1 only SAT branches. opt2 = 0 no trace on BIR/BDR. opt2 = 1 trace BIR/BDR.

e. Printer output

;J	Divert DELTA output to line printer.
a,b;0	Print hex dump from a to b on line printer. A header for the dump may be appended to 0.

f. Miscellaneous

e1,e2,v;Z	Store v in memory from e1 through e2.
;RK	Display addresses as CSECT type symbol plus any hex offset.

UTS BASIC AND BPM/BTM BASIC

The following differences are completely described in Chapter 5 of the BASIC/Reference Manual, 90 15 46 - Revision B or later.

1. Language extensions

a. String capability

String variables (scalars, matrices, substrings).
String expressions.
Character strings.
Length and value assignments (LEN, VAL).
Numeric-to-string conversion (STR).
Assignment and concatenation.
Comparison.
I/O.
String-to-alphanumeric constant conversion.

b. New intrinsic functions - CSC, SEC, COT, ASN, ACS, HSN, HCS, HTN, LTW, DEG, RAD.

c. CHAIN LINK statement

2. New edit mode commands

a. CLEAR

ARRAYS
STRINGS

b. NULL

ARRAYS
STRINGS
SIMVARS

c. FILE PACK

d. SET

e. EXECUTE

3. Increased edit mode-execute mode commands
   a. Changes in BREAK-PROCEED logic
   b. Changes in direct statement capability
      (1) Smaller number of non-direct statements
      (2) Direct capability in edit mode

UTS COUNTERPARTS TO FERRET COMMANDS

Most of the functions of the BTM FERRET subsystem can be accomplished with UTS TEL and PCL commands. The FERRET commands and their UTS counterparts are listed in Table C-3.

MISCELLANEOUS INFORMATION

Miscellaneous differences between BTM and UTS are listed below:

1. In UTS, read operations, through a DCB assigned to the NO operational label return an end-of-file code.
2. BTM and UTS load modules have different formats. Therefore, load modules currently running under BTM must be reformed under UTS before they will execute correctly. ROMs are compatible between BTM and UTS.
3. Under BPM/BTM, X'15' corresponds to a carriage return and X'25' corresponds to a line feed. Under UTS, X'15' corresponds to a line feed and X'0D' corresponds to a carriage return; X'25' is unassigned.
4. UTS does not set ASN in the DCB to 5 if the DCB is assigned to a Teletype as in BTM. UTS sets ASN to 3, DEVF to 1, and TYPE to 10. TYPE is not set until the DCB is opened. Prior to the opening of the DCB, it may contain an OPLB code or the EBCDIC representation of that OPLB.
5. In UTS, all input/output through COC routines (M:UC) is restricted to 140 characters.

Table C-3. FERRET Commands and Corresponding UTS Commands

FERRET Command	UTS Command	Description
>L[IST][acct]	!PCL <LIST [.acct]	Lists the specified account directory.
>T[EST] file	!PCL <LIST fid	Tests file accessibility.
>A[CTIVITY] file	(none)	Checks file activity.
>S[TATISTICS]	!STATUS	Displays accounting statistics for this on-line session.
>LOG		
>RAD		
>RADS		
>CPU		
>IO		
>SERV		
>S[TATISTICS] >N	!DISPLAY	Displays system load parameters.
>D[ELETE] file	!DELETE file	Deletes specified file.
>C[OPY] file$_1$,file$_2$ >K[OPY] file$_1$,file$_2$	!COPY file$_1$ to file$_2$	Copies file$_1$ to file$_2$. Copies file$_1$ to file$_2$ retaining keys.
>E[XAMINE] fid >I[NSPECT] fid	!COPY fid[TO ME]	Examines a file. Examines a file and displays keys.
>M[ESSAGE] text	!MESSAGE text	Sends message to operator.
>P[UNCH] fid	(none)	Punches file to paper tape.
>G[RANULES][(acct)]	(none)	Displays number of granules.
>R[EVIEW]	!PCL <REVIEW fid$_1$[,fid$_2$]	Lists and selectively keeps or deletes all files in account.

INDEX

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.

E

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.

F

G

H

I

L

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.

M

N

O

P

Q

R

S

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.

T

U

W

X

Xerox Data Systems

XEROX®

READER COMMENT FORM

We would appreciate your comments and suggestions for improving this publication.

Publication No.	Rev. Letter	Title	Current Date

How did you use this publication?

☐ Learning ☐ Installing ☐ Operating
☐ Reference ☐ Maintaining ☐ Sales

Is the material presented effectively?

☐ Fully covered ☐ Well illustrated
☐ Clear ☐ Well organized

What is your overall rating of this publication?

☐ Very good ☐ Fair ☐ Very poor
☐ Good ☐ Poor

What is your occupation?

Your other comments may be entered here. Please be specific and give page, column, and line number references where applicable. To report errors, please use the XDS Software Improvement or Difficulty Report (1188) instead of this form.

Thank you for your interest.

Fold and fasten as shown on back.
No postage needed if mailed in U.S.A.

Your name and return address.

STAPLE

STAPLE

FOLD

FIRST CLASS
PERMIT NO. 229
EL SEGUNDO, CALIF.

BUSINESS REPLY MAIL

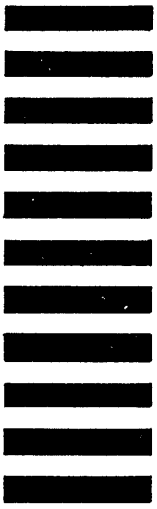
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

Xerox Data Systems

701 South Aviation Boulevard
El Segundo, California 90245

ATTN: PROGRAMMING PUBLICATIONS



CUT ALONG LINE

FOLD

Xerox Data Systems

XEROX®

701 South Aviation Boulevard
El Segundo, California 90245
213 679-4511