August 1978

This manual provides an overview of the TRAX system. This overview includes a general system description, a sample terminal session, and specific information on various TRAX system features.

Introduction to TRAX

Order No. AA-D327A-TC

OPERATING SYSTEM AND VERSION: TRAX Version 1.0

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard. massachusetts

First Printing, August 1978

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

Copyright © 1978 by Digital Equipment Corporation

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10

TABLE OF CONTENTS

PREFACE

This manual introduces the concepts of the TRAX Transaction Processing System, and describes the operation of a TRAX transaction processor to provide a general background for reading other manuals of the TRAX document set.

The intended audience for this manual ranges from a nontechnical manager who is considering installing a transaction processing system, to an application designer who needs to understand the system in detail.

If you want a brief introduction to the TRAX system, read Chapters 1 and 2. If you need further detail on TRAX system operation, read Chapters 3, 4 and 5.

The following list summarizes the content of each chapter.

Chapter	Content
Chapter 1 – Introduction	Provides an overview of the TRAX system.
Chapter 2 – Sample Terminal Session	Provides a sample terminal session running a transaction processor to illustrate the end product of the system.
Chapter 3 – TRAX System Operation	Describes the various elements of a transaction processor and how they interrelate. Describes the program flow within a transaction processor and describes which elements control it.
Chapter 4 – Special TRAX Capabilities	Describes a number of special features of the TRAX system, such as staging and journalling.
Chapter 5 – The Support Environment	Introduces the nontransaction processing capabilities of the TRAX system. Transaction processors are developed in the support environment.
Appendix A – TRAX Users and the Manual Set	Describes different users of the TRAX system and the different manuals in the TRAX manual set.
Appendix B – Programming Examples	Provides actual examples of the coding for specific forms and Transaction Step Tasks.
Appendix C – Glossary	Provides a glossary of key TRAX terms.

CHAPTER 1

INTRODUCTION

Many computer applications require on-line access to a data base. For example, when you call a catalog office to check on your order, it is vital that their representative be able to find your record quickly to verify delivery. The catalog office representative might also have to check inventories if the pertinent order involved is a hard-to-get or backordered item. Such an on-line access to a data base to add, delete, change, or verify information is an example of a transaction.

In TRAX, a number of terminal users can simultaneously access the data base from their respective terminals, and each user appears to have sole access to the data base. Further, any changes to a data file are made immediately, rather than being held for later processing.

Another characteristic of TRAX is the use of predefined forms. Forms are the structured arrangement of data on a video terminal, or occasionally, on a hard-copy terminal. For example when your catalog order is displayed on the representative's screen, its data is formatted so that your name always appears in one place on the screen, your address in another, and so on.

1.1 WHAT IS A TRAX TRANSACTION?

A transaction consists of all the processing steps required to perform one logical data-processing operation. For example, an order entry transaction might consist of a number of interactions between the system and the terminal user, but all of these interactions are part of the overall operation of entering an order. All together, these interactions or exchanges represent one transaction.

1.1.1 Transactions and Exchanges

Transactions therefore, consist of one or more exchanges. Each exchange represents one complete interactive cycle with the user. Figure 1-1 illustrates a simple exchange, which starts with a form requesting data from the user. The user types in the data, presses the ENTER key and the data is processed by the application program. The application responds by displaying "TRANSACTION COMPLETE" or by displaying the results of the processing. The application also provides instructions to the user specifying the next step to take.

This communication cycle between the user and the system is called an exchange. The word exchange is used because this cycle represents a complete "exchange" of information between the user and the system. Figure 1-2 illustrates a transaction comprising three exchanges. To perform the transaction, the terminal user first selects Transaction A from a "menu" of possible transactions. The transaction then begins. A form appears on the terminal requesting data from the user. The user inputs the required data and presses the ENTER key. The data is then processed by the application. When the processing is complete, rather than responding to the user within the same exchange, control passes directly to the next exchange, Exchange 2.

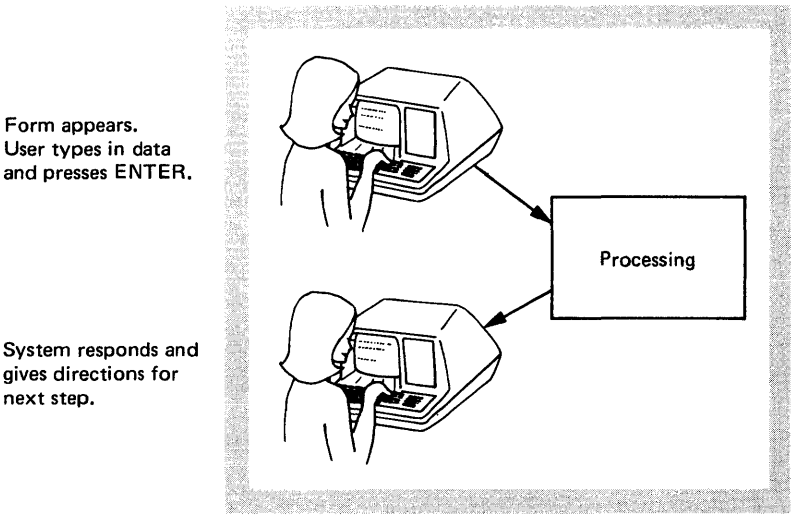


Figure 1-1 A Simple Exchange

Exchange 2 is performed in the same manner as Exchange 1, except that the data requested at the terminal and the processing required are both different. Notice that in both cases, after the user presses the ENTER key, a different form appears.

In Exchange 3, the same sequence of data entry and processing occurs. In addition, the system responds to the user by displaying a message such as "TRANSACTION COMPLETE", and by supplying directions for the next step.

1.1.2 An Order Entry Transaction

Figure 1-3 illustrates a typical TRAX transaction. The general format of the transaction is identical to that shown in Figure 1-2. The transaction consists of three exchanges, and is used to enter an order on-line. The transaction demonstrates that in TRAX a transaction represents the entire sequence of processing necessary to perform one logical operation: in this case, entering an order.

To run the transaction, the user first selects the Enter Order transaction and presses the ENTER key. The procedure for making the selection is described in Chapter 2. The system starts an Enter Order transaction beginning with its first form, shown in Exchange 1. The user types in either customer number or customer name, as desired, and presses ENTER. The application finds the record for the customer in the customer file and retains the information for use in Exchange 3.

The form shown in Exchange 2 appears after the user presses ENTER in Exchange 1. The form requests certain general information regarding the order such as purchase order number, and identity of the order taker. The user types in this data and again presses ENTER. The application holds the data for Exchange 3.

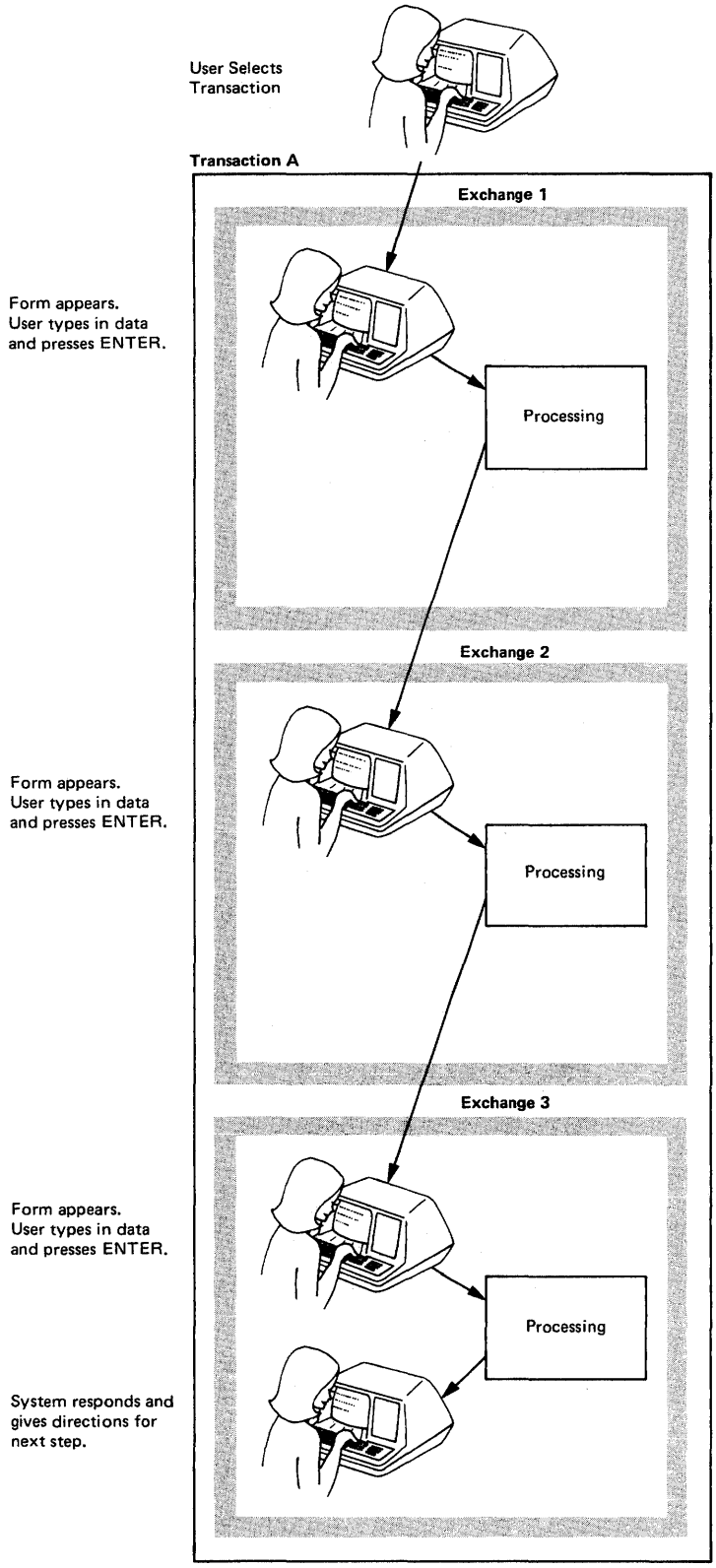
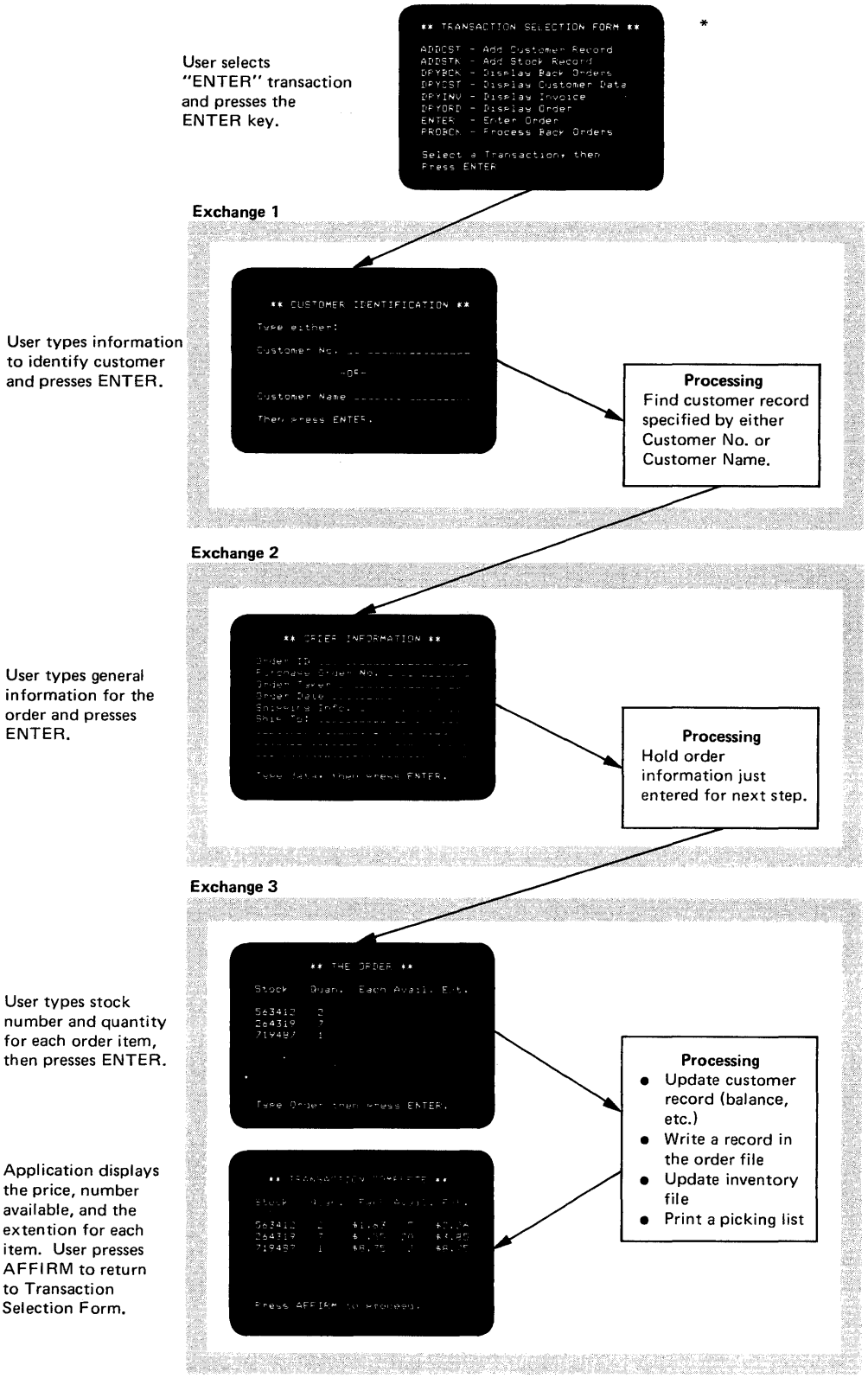


Figure 1-2 A Multiple Exchange Transaction



*All forms shown are mock-ups for illustrative purposes.

Figure 1-3 An Order Entry Transaction

Exchange 3 begins with a form which requests the specific line items of the order. In the form shown, the user has already typed in three such items, giving the stock number and quantity for each. When the order is complete, the user presses ENTER. The application then performs the necessary processing for the order. The application makes sure that inventory of each item is sufficient, and if not, displays an error message. The application totals the order, and makes sure that the customer's credit limit is sufficient. If not, it displays an error message. If the order is satisfactory, the application changes the customer's balance in the customer record, writes a record in the order file, changes the inventory file to reflect the items to be shipped, and prints a picking list for the stock picker. The application then modifies the form to show the price, quantity available, and extension for each item. Finally, the user presses the AFFIRM key to return to the original Transaction Selection Form.

Notice that each exchange uses a different form. This is another characteristic of an exchange: that it may have only one form. The form may be modified (as in Exchange 3), but a totally different form may not be used.

1.1.3 Structure of the TRAX Transaction Processing System

The TRAX Transaction Processing System provides a complete support structure for creating and running a transaction processor. A transaction processor is the software used to run one or more transaction processing applications. In addition, the TRAX Transaction Processing System provides a number of services for other non-transaction processing functions. These are known as "support environment services." They include batch processing of data and a number of services such as creating, editing, compiling, debugging and running programs for users at "support terminals", which are terminals other than those used for transaction processing.

The general structure of TRAX is illustrated in Figure 1-4. Notice that the diagram is divided into two parts: the application environment and the support environment. The application environment contains up to two transaction processors, with their corresponding application terminals. A second transaction processor, which is allowed on certain system configurations can be used for testing improvements to the standard transaction processor.

The support environment supplies services both to batch processors and to application programmers working at support terminals. The TRAX kernel services both the application and the support environment. Trax is a multi-tasking system; it can support a large number of simultaneous tasks (or programs).

In a TRAX system, transaction processors have priority over support environment programs.

1.2 TERMINAL USER ORIENTATION

Operationally TRAX is designed to be easy for the terminal user. This ease of operation is achieved in three ways:

- The VT62 video terminal itself is easy to use.
- The forms that appear on the terminal screen can be patterned after paper forms which are normally familiar and accessible to the user.
- The communication between the user and the application is simple and straightforward.

Each of these topics is discussed below.

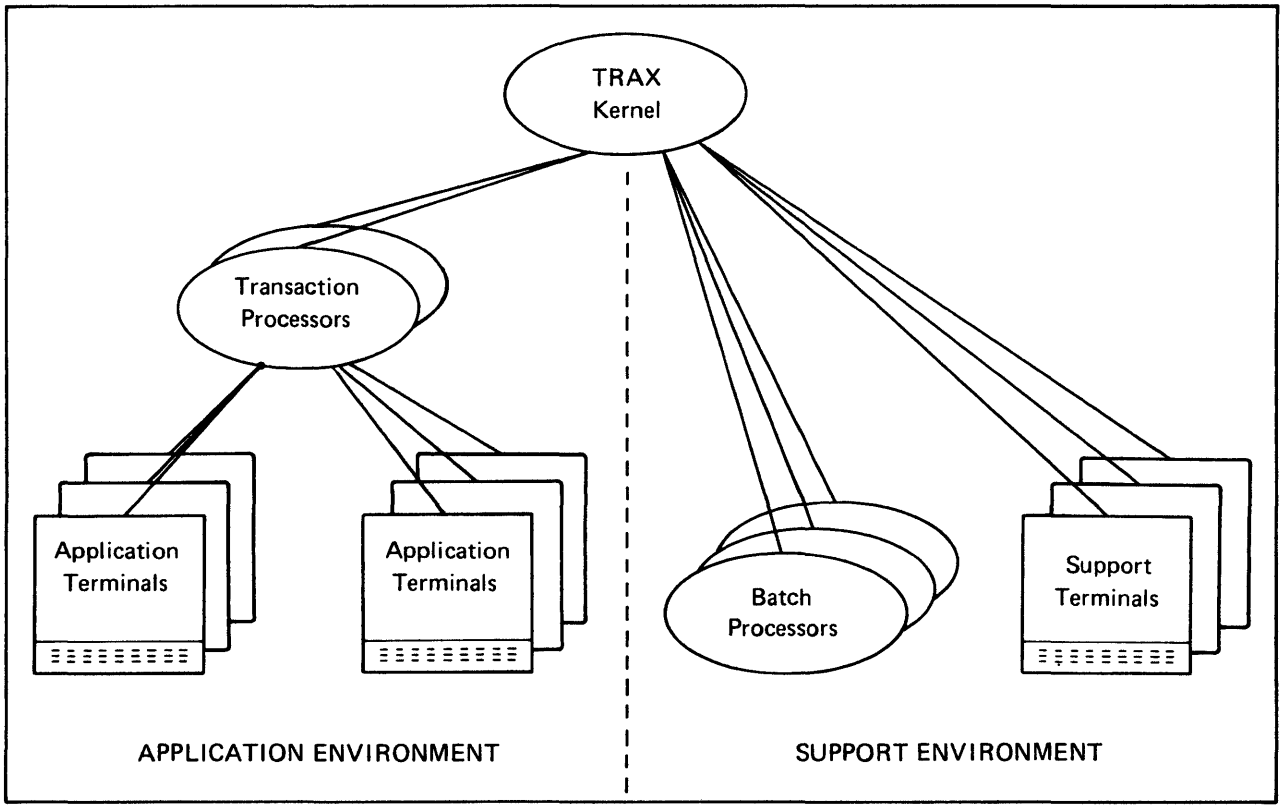


Figure 1-4 TRAX System Structure

1.2.1 The VT62 Terminal

The VT62 video terminal (shown in Figure 1-5) is used as the interactive application terminal in the TRAX system. The user actually performs transactions using this terminal. The VT62 contains a microcomputer that can perform a number of data processing functions within the terminal itself, thereby reducing the workload on the rest of the system. This results in instantaneous response to individual keystrokes. In addition, the VT62 provides a series of keys which simplify the process of data entry and perform special functions when used with a TRAX Transaction Processor. A few of the features of the VT62 terminal are:

Block Mode. The VT62 terminal is a block mode terminal, which means that it transmits all user-entered data at one time. In practice, a user can type in a complete screen of data, make any corrections, and then transmit the data to the system by pressing the ENTER key. Handling a complete screen of data in this way is faster for the user and creates a greater sense of control.

Reverse Video. The VT62 allows reverse video or black-on-white to be selected for individual fields on the screen. The reverse video field appears as a white rectangle against the normally dark screen background. A prime advantage of displaying fields in reverse video is that you can see how many empty spaces remain in the field, thereby simplifying data entry.

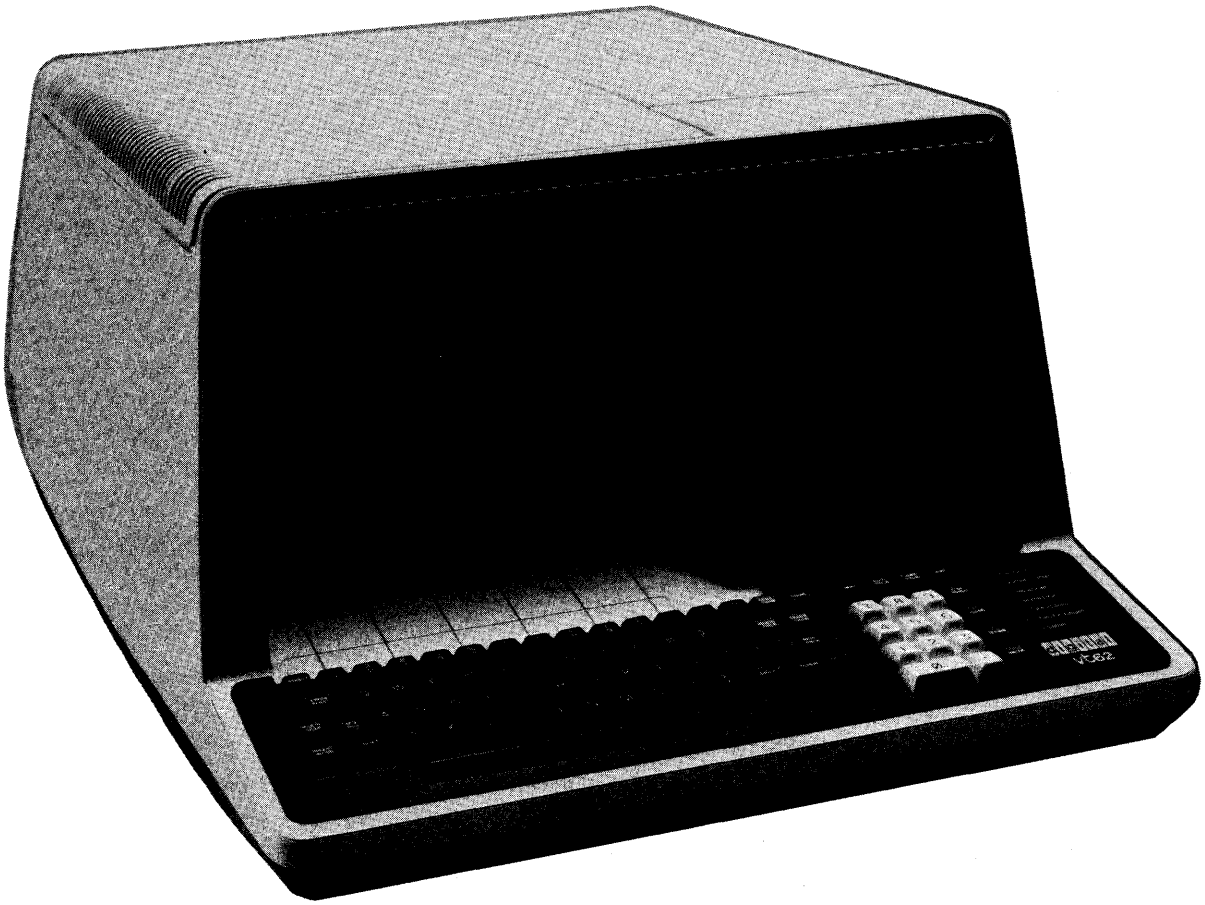


Figure 1-5 The VT62 Terminal

INSERT MODE and DELETE CHARACTER Keys. These two keys provide for direct editing of characters. Characters can be inserted or deleted at the location of the cursor (the flashing square) on the screen. This makes it easy to correct individual data items without having to retype a whole line.

Right-Justified Fields. The VT62 allows alphanumeric characters to be entered onto the field in the same way as if entered onto a pocket calculator, which is, right-justified or always aligned with the right border of the field. This avoids the necessity of having to type in leading zeros in a numeric field that is half full.

Numeric and Alphabetic Fields. The VT62 terminal allows any given field to be defined as numeric or alphabetic only. For example, if a terminal operator types a Y into a zip code field that is specified as numeric by the application programmer, the terminal beeps, the keyboard locks, and an error message appears on the screen saying "NUMERIC ONLY." To correct the error, the user must press the ERROR RESET key, and then typing can continue. This error correction sequence is performed by the terminal itself, and does not require application program assistance.

Other Features. The VT62 terminal has a number of other features. For example, there are a wide variety of ways of positioning the cursor on the screen. (See the *TRAX Application Terminal (ATL) Language Manual*). The VT62 also handles the Digital Data Communications Message Protocol (DDCMP) and can exist in a multi drop environment as well as point-to-point. Finally, the VT62 allows connection of an optional forms pointer without additional interfacing hardware.

1.2.2 TRAX Forms

TRAX forms are another system feature that make it easy for the terminal user to operate the system. A form is a structured arrangement of data that appears on the VT62 screen. The use of forms allows the user to input a complete screen of data at a time, rather than having to input it, piecemeal making it easier for the user to handle data.

One advantage of forms is that they allow data to be formatted on a video terminal in a way that duplicates existing paper forms. For example, a typical purchase order (a paper form) is shown in Figure 1-6. The implementation of this paper form into a TRAX form is shown in Figure 1-7. Notice that all fields on the form requiring user input are shown in reverse video.

1.2.3 Communication Between User and Application

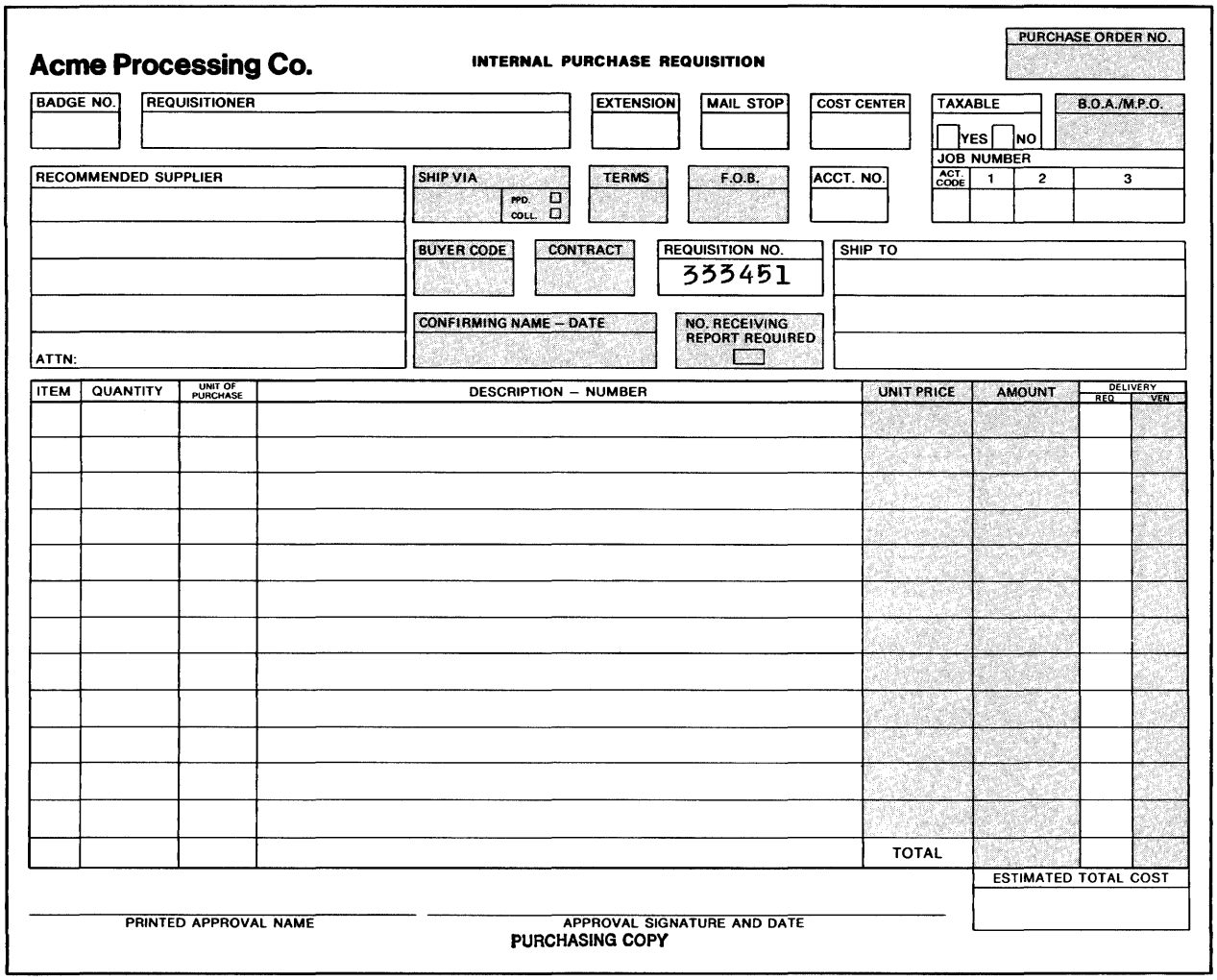
The TRAX system provides a series of function keys which allow for easy communication between the user and the application. These keys are:

- ENTER
- CLOSE
- AFFIRM
- STOP-REPEAT
- ABORT
- Certain other user-defined keys

These keys allow the user to control the flow through the exchanges of a transaction. For example, in the transaction example shown in Figure 1-8, the user first selects the DISPLAY transaction from the Transaction Selection Form. Form 1 of that transaction then appears to request the customer number or the customer name for the customer record to be displayed. The user then types in this data and presses ENTER. Optionally, the user could press CLOSE, which would cause the Transaction Selection Form to reappear. After pressing ENTER, Form 2 of the transaction appears, displaying the contents of the customer record.

At this point, the user has three options:

- Press ENTER to display the next sequential record in the file.
- Press AFFIRM to return to form 1 so as to enter a different customer number or name.
- Press CLOSE to return to the Transaction Selection Form to select a new transaction.



Acme Processing Co. INTERNAL PURCHASE REQUISITION

PURCHASE ORDER NO.

BADGE NO.	REQUISITIONER	EXTENSION	MAIL STOP	COST CENTER	TAXABLE ☐YES ☐NO	B.O.A./M.P.O.

JOB NUMBER

ACT. CODE	1	2	3

RECOMMENDED SUPPLIER	SHIP VIA (PPD. ☐ COLL. ☐)	TERMS	F.O.B.	ACCT. NO.

BUYER CODE	CONTRACT	REQUISITION NO.	SHIP TO
		333451	

CONFIRMING NAME – DATE	NO. RECEIVING REPORT REQUIRED ☐

ATTN:

ITEM	QUANTITY	UNIT OF PURCHASE	DESCRIPTION – NUMBER	UNIT PRICE	AMOUNT	DELIVERY REQ	DELIVERY VEN
				TOTAL			

ESTIMATED TOTAL COST

PRINTED APPROVAL NAME

APPROVAL SIGNATURE AND DATE

PURCHASING COPY

Figure 1-6 A Purchase Order (Paper Form)

Function keys allow the user to select the next operation to be performed. The actual keys used are determined by the application programmer following general system guidelines.

1.3 APPLICATION PROGRAMMER ORIENTATION

The TRAX system is designed to be an easy system in which to implement an application. The general structure of a transaction processor, and some of the system features which make it simple for the application programmer to create a transaction processor, are described below.

1.3.1 Transaction Processor Structure

The general structure of a transaction processor is shown in Figure 1-9. A transaction processor consists of a combination of user-written elements and system-supplied elements.

A TRAX system consists of a kernel which performs the normal operating system peripheral management functions, one or more transaction processors, application programs, and support programs. The transaction processor consists of the TRAX Transaction Processor Executive, application program elements defined by the user, and a dynamic buffer area called the software cache.

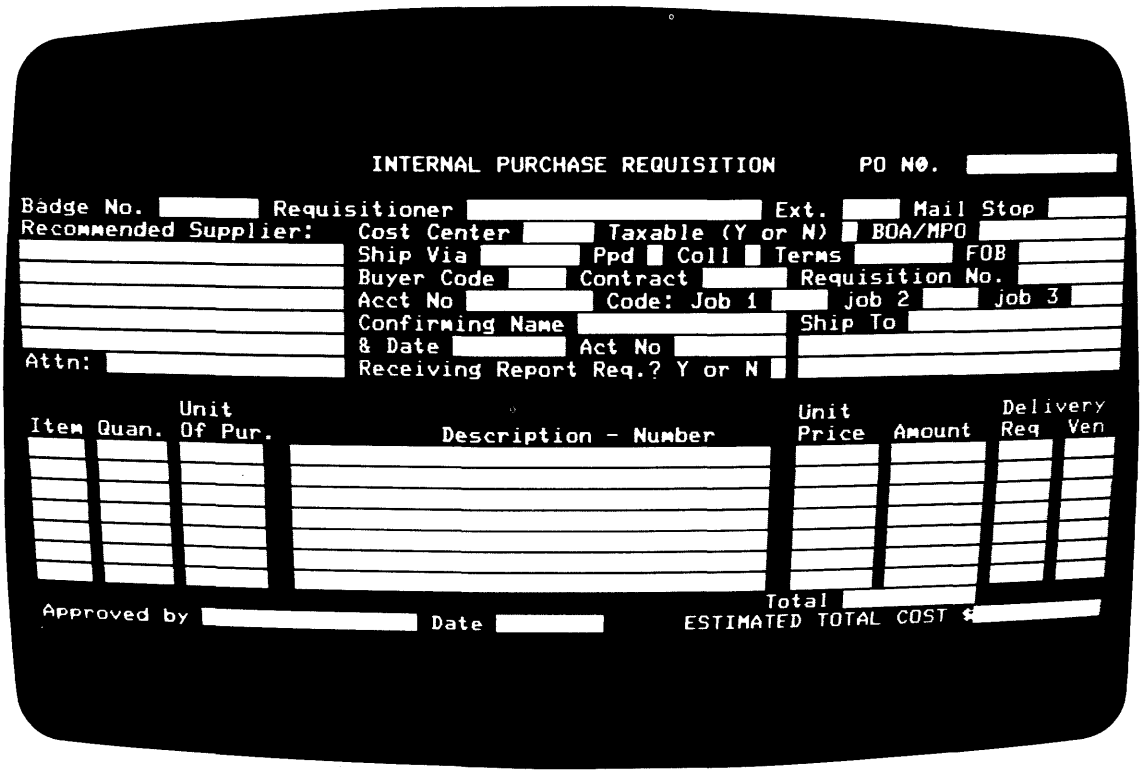


Figure 1-7 A Purchase Order as a TRAX Form

The Transaction Processing Executive performs a wide range of services for the transaction processor such as file management and application flow control. If not provided, such functions would have to be specially programmed by the user for the particular application.

The application program elements defined by the user fall into three categories:

- Transaction Step Tasks
- Forms Definitions
- Definition Files

Transaction step tasks (TSTs) are the specific user-written program segments necessary for the system to perform the processing for the pertinent application. TSTs may be written in COBOL, or BASIC-PLUS-2 in the form of subroutines.

Forms definitions are file records that define the exact appearance of the forms on the applicative terminals. Forms definition records specify all possible forms modifications which might be required for such purposes as sending error messages to the user, or providing directions for the next user operation.

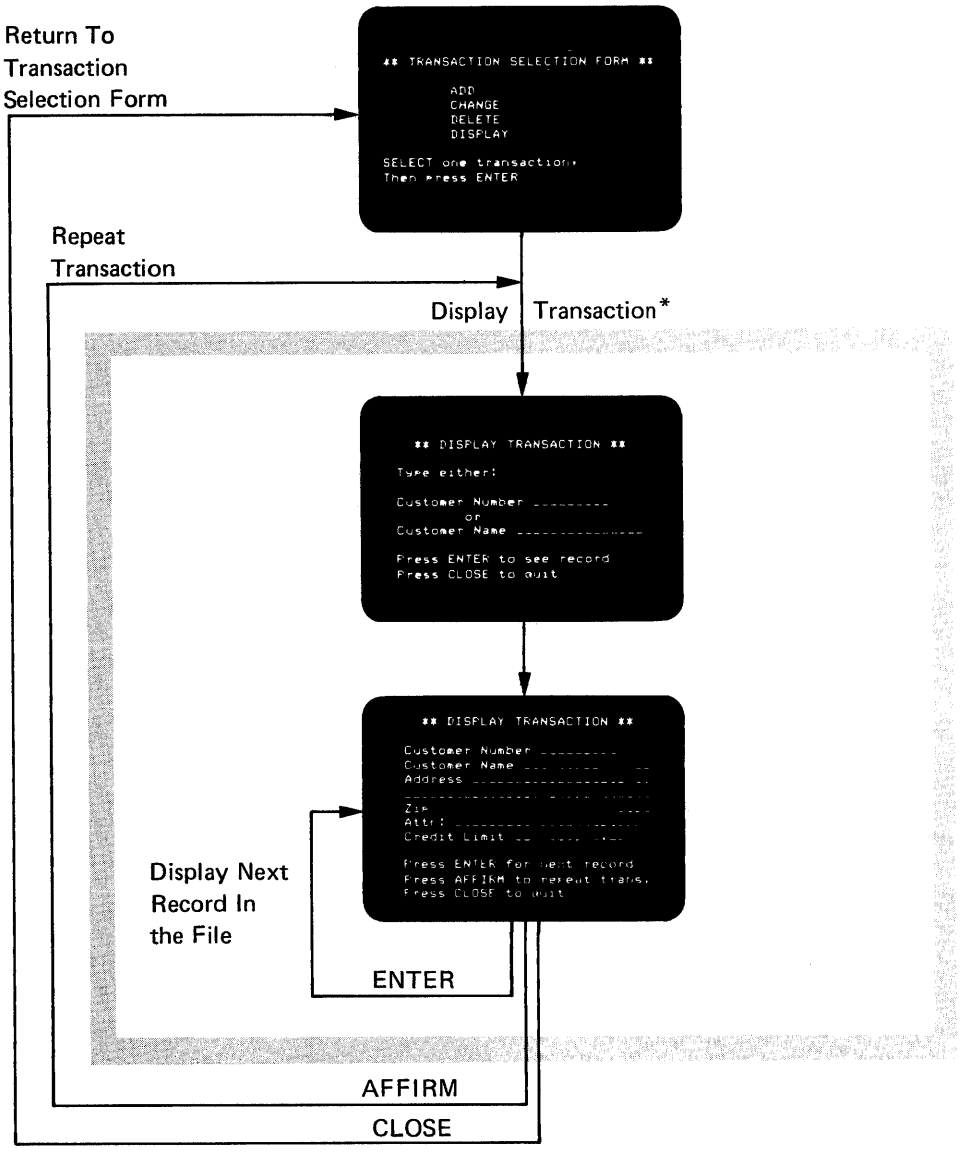


Figure 1-8 Effects of Function Keys in a Sample Transaction

In addition to transaction step tasks and forms definitions, a transaction processor is defined by a series of user-created entries in several files. These definition files are created by the user through use of interactive utilities. The files perform such functions as specifying the interaction between different transaction step tasks and the different forms through which data is displayed and entered at an application terminal. Definition files serve to separate control from procedure and assure simplified system maintenance and modification.

1.3.2 COBOL and BASIC-PLUS-2

The TRAX system provides support for user-written applications. These application programs are written in COBOL or BASIC-PLUS-2. The ANSI 74 COBOL provides the user with a recognized standard language for data processing applications. BASIC-PLUS-2, an extended version of the

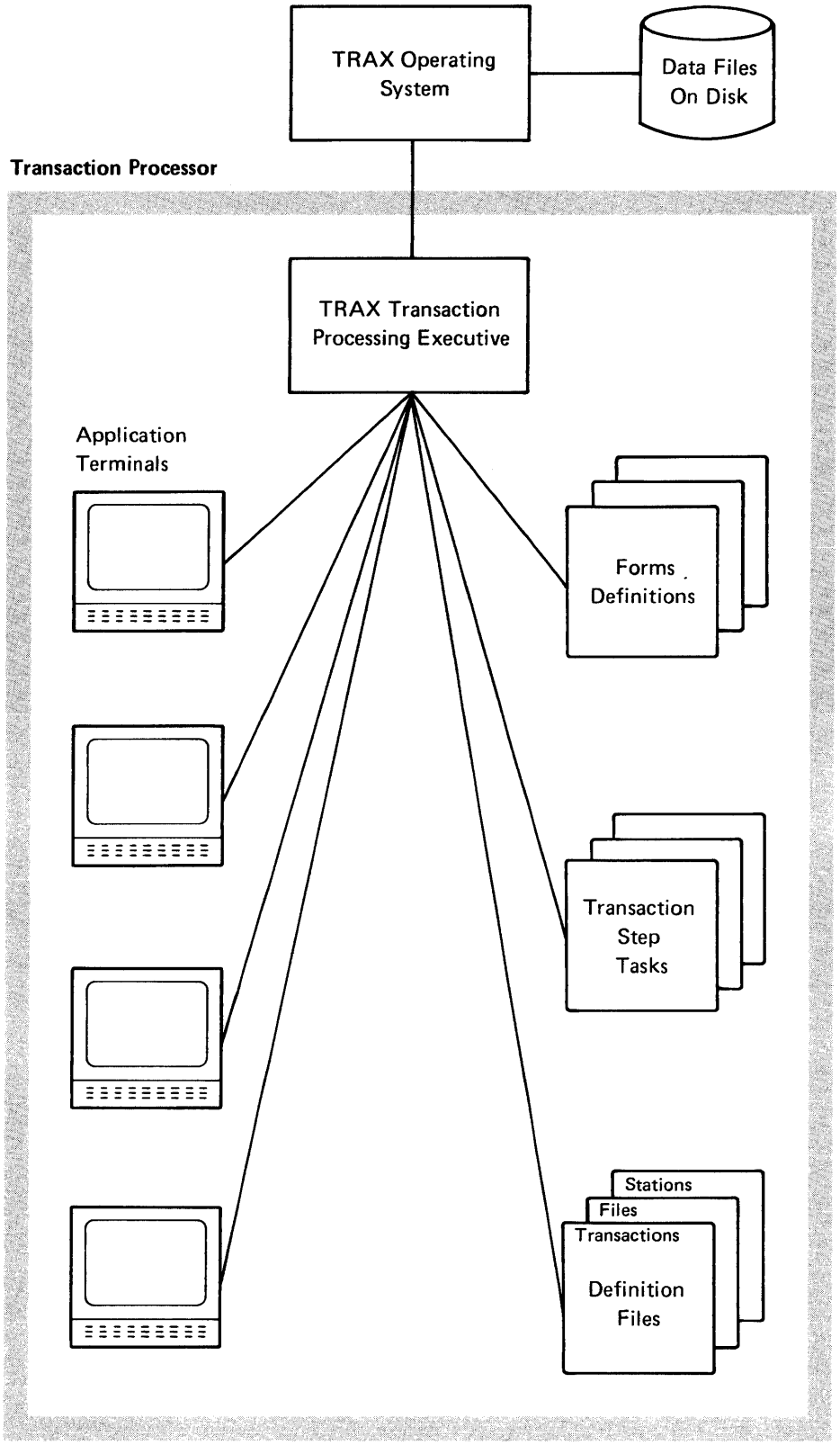


Figure 1-9 Transaction Processor Structure

BASIC language, provides the user with a language that is easy to program and that provides a number of features for commercial and data-record oriented applications.

Because the systems-level programming is provided in TRAX, the user can write application-specific code in these higher-level languages without fear of poor efficiency and slow execution time. The user gains all the benefits of writing in higher-level languages: no retraining of programmers, faster coding, ease in understanding programs, and simpler maintenance.

1.3.3 Transaction Step Tasks (TSTs)

The user writes the code in COBOL or BASIC-PLUS-2 for a specific application in the form of short subroutines, or program sections, called transaction step tasks (TSTs). A given exchange in a transaction contains one, or more, TSTs. As illustrated in Figure 1-4, the data from one exchange can be processed by several TSTs.

A TST normally represents one logical step in the processing. For example, the function of a given TST may be to read one record from a file.

Dividing a program into a series of TSTs provides distinct advantages to the application programmer. Each TST can be coded and debugged separately. Since each TST is a logical unit of processing, it becomes easier to understand the operation of the application as a whole by examining the interrelationship of these logical units.

1.3.4 ATL, The Forms Language

In the TRAX system, the forms displayed on the terminals are not part of the application code (the TSTs), but rather are coded and stored separately. TRAX provides a forms language called Application Terminal Language (ATL), which is used to specify the structure of the forms. Using English words and simple formatting rules, the application programmer can quickly code any given form.

ATL allows the programmer to specify the initial appearance of the form as it is displayed on the terminal along with any possible modification to that form such as error messages or directions to the terminal user concerning the next step to take. ATL allows the application programmer to specify certain attributes of each field, such as whether a field is to be displayed in reverse video or whether it is a field into which the user can input data. ATL also allows the programmer to specify which function keys are "enabled" or "disabled" at a particular point in user/application interactive dialogue. If a key is enabled, pressing it will produce the appropriate response for that particular key. If a key is disabled, pressing it will produce no response at all.

To create a form using ATL, the application programmer first creates a source file containing the proper ATL coding. The programmer then compiles this file using the ATL utility program. The ATL utility generates a listing which provides a variety of information about that particular form. The ATL utility also creates a record representing the form in a file called the Forms Definition File. When the transaction processor is running, the Executive refers to this file to find the information it needs to display a particular form.

For the application programmer, use of the ATL forms language simplifies the task of creating a transaction processor. Coding for the forms is performed separately from the procedural coding for the application. As a consequence, it is easier to make modifications to the forms later to improve their usefulness. Also ATL performs error checks which simplify the process of correctly coding a form.

1.3.5 Definition Files

After coding the forms and the transaction step tasks (TSTs) for a particular transaction, the application programmer uses a series of interactive utilities to define the interrelationship of the various components for that transaction. These definitions are stored in definition files which are then used by the system to control such things as the sequence of execution of the different TSTs in a transaction.

Three important definition files include the transaction definition file, the file definition file, and the station definition file.

> The transaction definition file contains definitions of all transactions for a given transaction processor, including the name of each exchange in the transaction, the name of the form for each exchange, and the sequence of TSTs for each exchange. Functionally, this file defines how all the components of a transaction interrelate. The file definition file contains a definition of each data file in the application, including such definitions as record size, logical name, and file organization. The station definition file provides a definition of each logical element in the system such as a terminal or a TST. Stations are described in further detail in Chapter 3.

A sample listing of a terminal session with one of the interactive utilities is shown in Appendix B. The utility of interest produces the transaction definition file. Those items which the user inputs are underlined. If the user wants more information concerning a question asked by the system, he can type a question mark. The values enclosed in angle brackets (<>) indicate the default values for a question if the user types only a carriage return. For a detailed explanation of this utility, see the *TRAX Application Programmer's Reference Manual.*

Use of these definition files in the TRAX system provides several advantages. For example when it is necessary to change the system, you need only change the corresponding definition file. In addition, to add two more terminals to a transaction processor, the application programmer need only add the terminal station definitions in the station definition file. Also, the transaction definition file provides an overview of a transaction processor's organization which simplifies the job of maintaining the transaction processor.

1.3.6 Steps in Implementing a Transaction Processor

The process of implementing a transaction processor (Figure 1-10) consists of five steps: design, implementation, unit debug, transaction debug and production.

Design

This initial phase consists of translating the business problem to be solved into transactions and data files. Next each transaction is broken down into forms to be displayed and TSTs to process the entered data.

Implementation

Source files for forms and TSTs are written and entered into TRAX system via the TRAX interactive editor EDT.

Forms are compiled utilizing the ATL utility and TSTs are compiled utilizing the appropriate language processor, COBOL or BASIC +2.

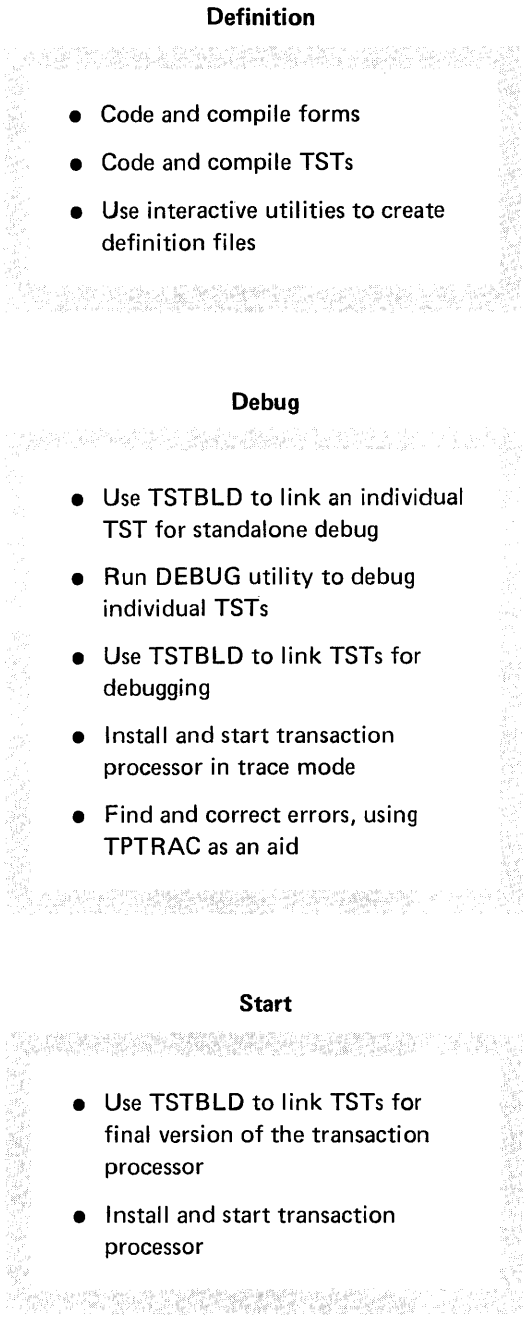


Figure 1-10 Steps in Implementing a Transaction Processor

Creation of the definition files for such elements as transactions, stations, forms, and files, is performed by running the TRAX definition utilities and answering the questions asked by each utility.

Unit Debug
Before TSTs can be executed, they must be built into tasks. This process is performed through use of the TRAX utility TSTBLD.

After being built, TST's are individually debugged in the support environment utilizing the TRAX TST debugger utility DEBUG. Once the logic of each TST has been tested, the TSTs can be integrated into a transaction processor.

Transaction Debug
At this point, a transaction processor must be built. This involves:

1. Building TSTs for transaction debug, using TSTBLD.
2. Installing and starting the transaction processor via the TPCTRL utility.

Each transaction is now tested utilizing an application terminal. The tools available for transaction debug include:

1. A TRACE feature which will log each operation performed and provide a formatted report.
2. A software error logger which records all errors caused by improper TST operations. A formatted report is provided.
3. Inclusion of a language debugger, with direct access through a support terminal.
4. Ability to set the entire data base to read only.

Production Use
To prepare for production use, the TSTs must be rebuilt using TSTBLD and the transaction processor must be re-installed without use of debugging features.

1.4 COMMUNICATION(S)
An integral part of the TRAX design is the ability to distribute the system resources. Because terminals are multi-drop and utilize Digitals standard communication protocal DDCMP, they can be clustered on a single communication's line and placed where needed.

The system through its TRAX/TL transaction link allows multiple TRAX systems to form a distributed data processing network, thereby placing the processing power where it is needed. In addition, the TRAX/TL-3271 allows a one or more TRAX system to be linked to IBM system running the CICS transaction processor monitor.

A different aspect of communications is the ability to communicate between support environment and application environment. This TRAX supplied feature is described in Section 1.4.3.

1.4.1 TRAX/TL Data Communication
In a TRAX/TL link, two or more TRAX systems may communicate with each other so as to share resources, especially their data files. For example, Figure 1-11 illustrates three TRAX systems, one in Los Angeles, one in Chicago and one in Boston. Each is used for processing transactions at its

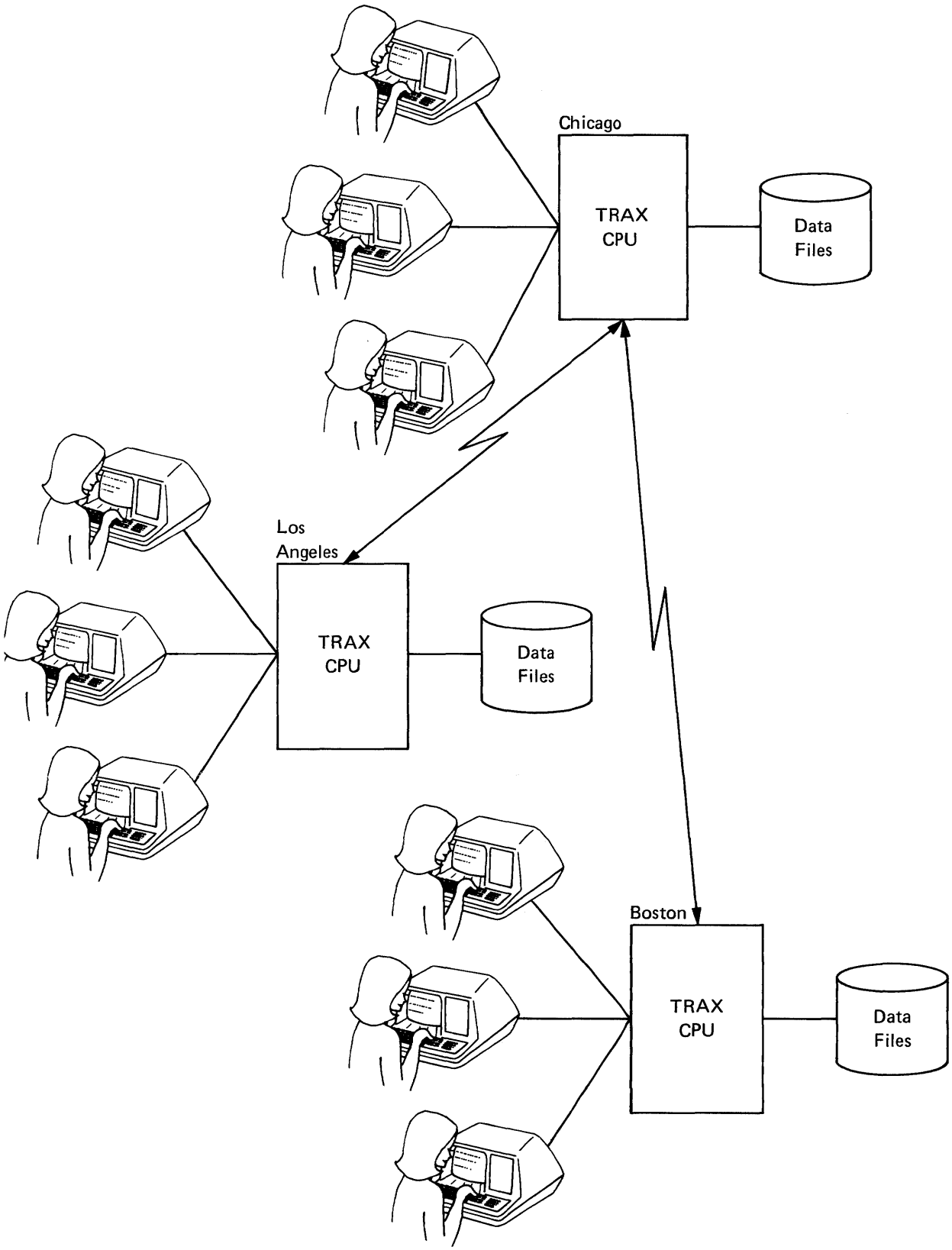


Figure 1-11 TRAX to TRAX Data Communication

respective location. However, from time to time each system requires access to data files from another system. To handle this requirement, the Los Angeles system has a data communications link with the Chicago CPU and a link with the Boston system. These links allow any system to access any set of data files in either of the other systems. For example, the Los Angeles transaction processor can initiate a remote transaction which accesses the Boston or Chicago data files as if it were a local transaction. In effect, a transaction processor in a given TRAX system can initiate a remote transaction to access the data base or other resources of any other TRAX system to which that system is connected.

1.4.2 TRAX-to-IBM Data Communication

A TRAX-to-IBM link might be used to place processing capabilities near the users, as in the previous example, or it might be used to provide additional transaction processing capabilities to an already existing IBM system. Optionally both might be combined.

Figure 1-12 illustrates a typical TRAX-to-IBM link. The TRAX system is a fully operational transaction processing system, complete with its own data files. In addition, it has a data link to an IBM system running CICS. The two CPUs may be in the same room or across the country. Other TRAX systems could be linked in as well.

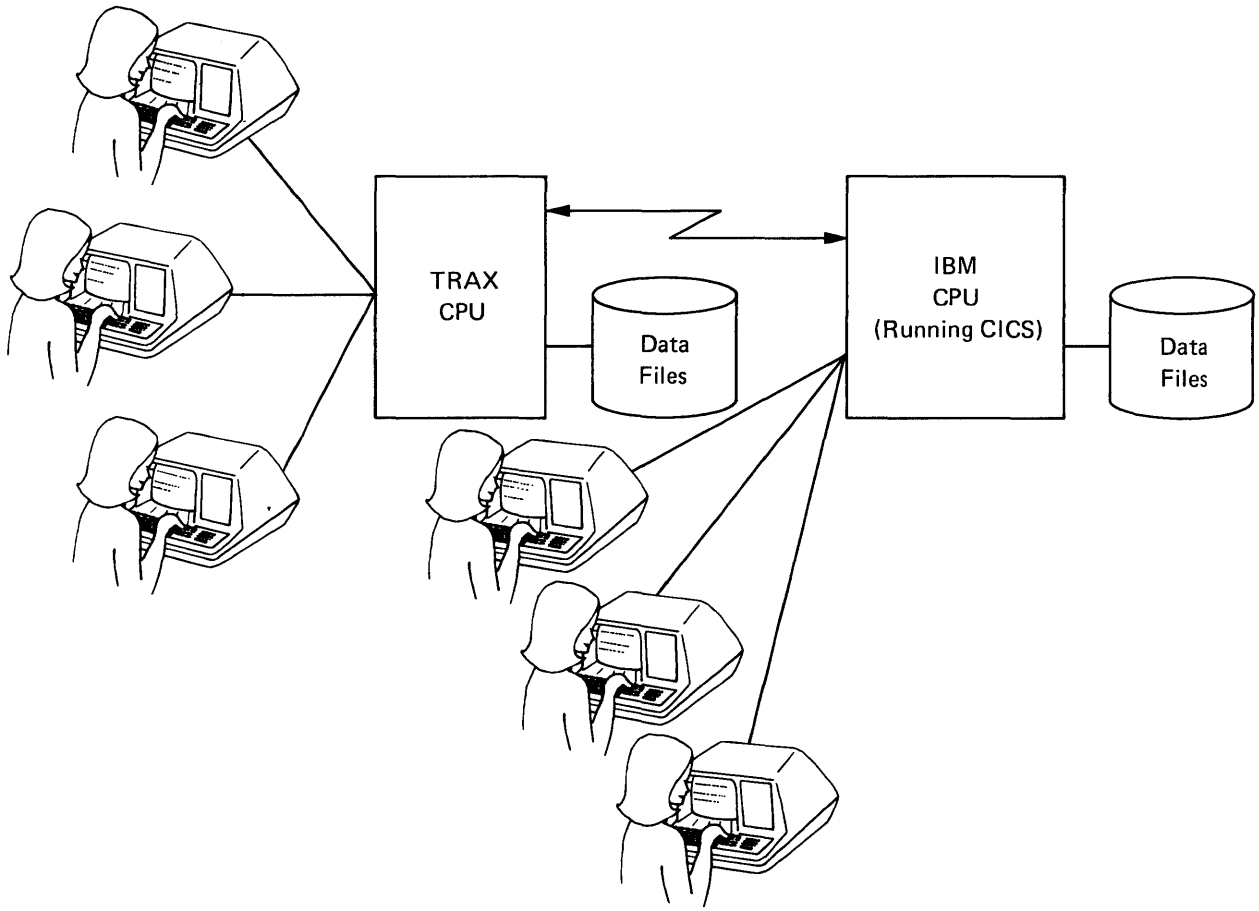


Figure 1-12 TRAX to IBM Data Communication

To the IBM system, a TRAX system appears to be set of terminals. Under this arrangement, the TRAX system can perform transactions as if they were being performed on the IBM machine itself.

1.4.3 Communication Between the TRAX Application and Support Environments

Communication between a TRAX system application and support environment involves batch processing and initiation of a single exchange transaction. In the application environment, an application program can be designed to allow the terminal user to submit a batch job for subsequent processing in the support environment. In the support environment, any program operating in that environment can initiate a single exchange transaction to be executed for that program.

The ability to submit a batch job from the application environment allows the processing of large or complex tasks without tying up the user terminal. It also facilitates maintenance of the TRAX data base.

Initiation of single exchange transaction from the support environment allows the use of existing application environment programs by programs in the support environment. The result is greater system efficiency as well as optimum use of system resources.

1.5 SYSTEM PERFORMANCE

TRAX is designed to provide high performance in transaction processing applications, in terms of both response time and rate of transaction throughput. Some of the system features which contribute to TRAX performance are discussed below:

- A data management service (RMS) which is both resident and shareable.
- Software Cache
- Use of Micro Processor Controlled Terminals
- Optimized System Code
- "Stationary" Messages

1.5.1 Data Management Services (RMS)

The TRAX data management service (RMS) provides a set of general purpose file handling capabilities that allow user-written application programs to create, access and maintain system data files in an efficient and timely manner. With RMS, you have a choice of implementing sequential, relative and indexed file structures.

Indexed files are reorganized incrementally when new records are inserted, therefore, insertions can be made on-line without off-line reorganization. Three methods of key searching are provided; exact, generic, and approximate.

Record access and retrieval can be done sequentially or randomly and records can be accessed concurrently. Provisions are made for efficient use of the file space as well as for high speed accessing of files.

Finally, RMS is equipped with language interfaces for both COBOL and BASIC-PLUS-2. Also comprehensive set of utilities are furnished to support efficient file transfer.

1.5.2 Software Cache

To reduce the number of times TRAX must read from the data files on disk, the system maintains a software cache in memory (Figure 1-13). Each time the system transfers a data block from disk

to memory, it is stored in software cache. Before transfering a block from disk, the system checks to see if that block is already in the cache. If so, the block is accessed directly from memory, and a disk operation is eliminated.

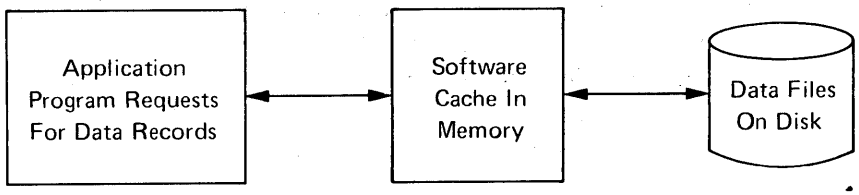


Figure 1-13 Operation of Software Cache

All data stored on disk is cached in this way. Caching applies not only to the user's data files on disk, but also to TSTs and forms definitions as well.

Experience shows that a significant percentage of blocks will be in the cache when needed. The size of the cache is user-determined. Generally all extra available memory is used for the cache. The greater the cache size, the greater the likelihood that a given block will be in the cache, thereby improving system performance.

1.5.3 Use of Microprocessor Controlled Terminals

One factor in the high performance of TRAX systems is the use of the VT62 terminal which operates under control of an internal microprocessor. The terminal user performs all data entry and editing operations at the terminal. Only after making changes and verifying the data entered, is the data sent to the TRAX system. Since the VT62 is a block mode terminal, the data is sent as a block rather than a smaller data units thereby providing for more efficient processing.

1.5.4 Microprocessor Communications Controller

All communications lines between application terminals and the TRAX system are controlled by microprocessors (KMC-11); thereby reducing the load on the host system.

1.5.5 Optimized System Code

In TRAX systems, the systems programming is supplied as shared routines. These shared routines which perform the various system management and application support functions are resident in system memory and are directly available to all application programs as needed. As a result, the application, developer need only prepare those parts of a program which are application specific.

1.5.6 "Stationary" Message

TRAX is a message-oriented, system where basic system execution depends on the transmission of messages from terminals to TSTs and back to terminals. In TRAX, after messages have been received by the terminal and formatted, none of these messages are actually moved within the system. Rather, only pointers showing the location of the messages are moved. The result is greater system performance.

1.6 DATA INTEGRITY

Maintaining the security and accuracy of the user's data files is a prime design objective of the TRAX system. There are three main system capabilities which promote data integrity:

- Data Backup and Journaling
- Maintaining Integrity of Data Records
- Security Provisions

1.6.1 Data Backup and Journaling

In transaction processing, a prime concern is the ability to recover from a disk failure without loss of data. In TRAX this is accomplished through a combination of disk backup and journaling.

The TRAX system provides a command that initiates copying of the user's data base onto another disk or to magnetic tape so that the user's disk files are "backed up". This backup procedure can be performed as frequently as desired, once a day being the general case. In the event of a disk failure, another command is available to restore the data base to its state as of the last backup.

In TRAX the problem of restoring transactions processed since the last back up function was performed can be solved by journaling. When journaling has been specified, the system writes a record onto magnetic tape or other medium for each change to the data base. These records can then be used to reconstruct the data base up to the time of disk failure. The system provides a utility program to perform this function.

1.6.2 Maintaining Integrity of Data Records

TRAX provides two system features; logical record locking and staging for maintaining the integrity of individual records in the set of data files.

Where two application-terminal users update the same record simultaneously, the effect of at least one of the updates would be lost. Locking is the system's primary means of protecting files from information loss due to simultaneous updates. A logical record requested by a TST can be locked in order to maintain data file set integrity by preventing simultaneous updates.

Staging is a system feature which, when specified, causes the system to retain all updates to the data base until a transaction has been completed. At that point, all changes are made at once. However, if in the middle of the transaction the terminal user changes his mind and decides to terminate the transaction, none of the updates are made. This insures that the data base is never left in an "in between" state with some records for a given transaction changed and others not.

1.6.3 Security Provisions

TRAX provides several features that make sure that only authorized persons can perform transactions or otherwise alter the data base. For example, application terminals are kept separate from support environment terminals so that it is impossible to gain direct access to the system from an application terminal.

Also, only certain user-preassigned transactions can be performed at a given application terminal. In addition, users can sign on, and after supplying a correct password can perform only the transactions for which they are authorized. In this way, it is possible to maintain rigid control of what persons or terminals have access to what transactions.

CHAPTER 2

A SAMPLE TERMINAL SESSION

One of the best ways to understand transaction processing under TRAX is to run through a sample application at a terminal. In demonstrating this sample application, it is assumed that you are actually sitting at a terminal and running transactions.

All TRAX applications are written by the user. The sample application described is an example of the type of program that you might write for your own application. Many features of the sample represent the way the application programmer chose to code this application and do not represent general TRAX features.

This sample application maintains an on-line file of customer records, each containing customer number, name, address, telephone number, "attention" line, and credit limit. This application can be used to add, delete, change, and display a customer record.

2.1 THE TRANSACTION SELECTION FORM

Generally the first form to appear on a screen is a Transaction Selection Form. The Transaction Selection Form for this transaction processor is shown in Figure 2-1. Throughout the sample session, the terminal used is the VT62.

A given application may consist of a large number of transactions. In this sample application there are four transactions:

- Add customer to master file
- Change customer file record
- Delete customer from master file
- Display customer file record

The Transaction Selection Form allows you to choose one of these transactions.

The structure of this transaction processor is shown in Figure 2-2. Notice that there are two levels: you are at either the level of selecting a transaction or at the level of performing the selected transaction. In order to perform another transaction it is necessary to exit from the current transaction and return to the Transaction Selection Form.

In selecting the transaction, note that the cursor (see Figure 2-1) is initially on the "A" of ADDCUS, the "add customer" transaction. When you press the NEXT FIELD key (see Figure 2-3) the cursor moves to the initial "C" of CHGCUS, and when you press the NEXT FIELD key again, the cursor moves to the initial "D" of DELCUS. If you want to select the DELCUS delete customer from master file, transaction, you have to press the SELECT key causing the name DELCUS displayed in reverse video, as shown in Figure 2-4. If you press the ENTER key, the first form of the delete customer transaction appears.

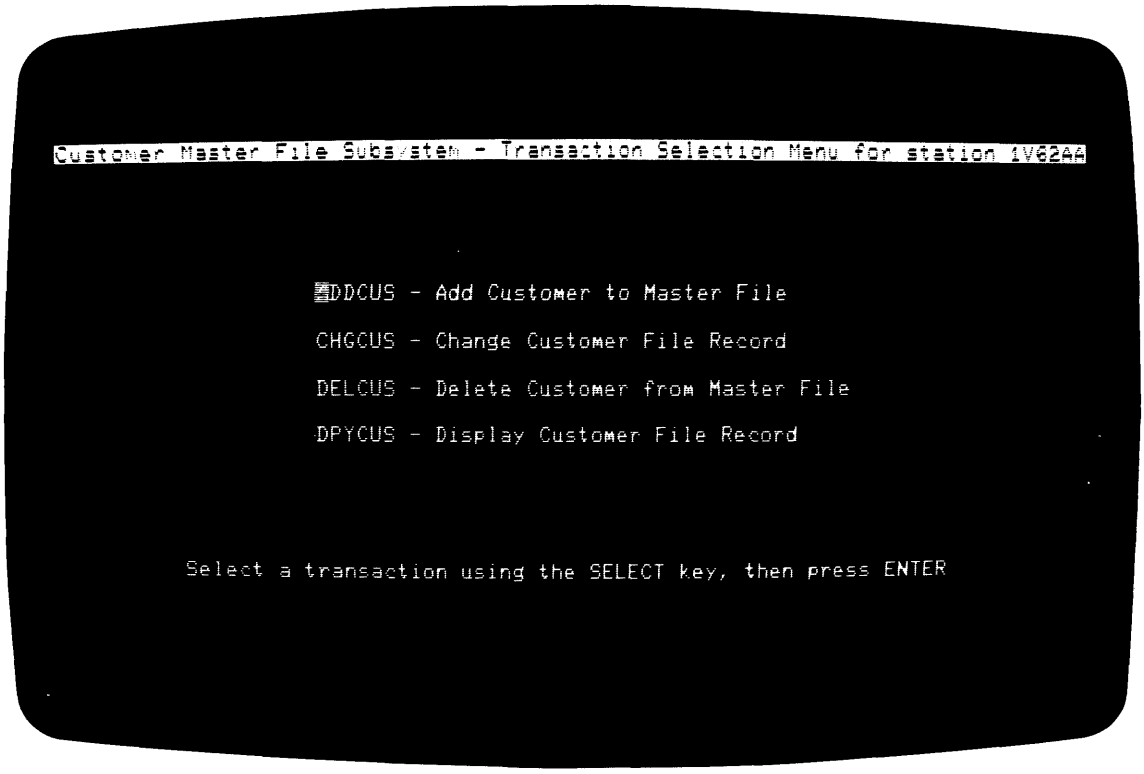


Figure 2-1 Transaction Selection Form

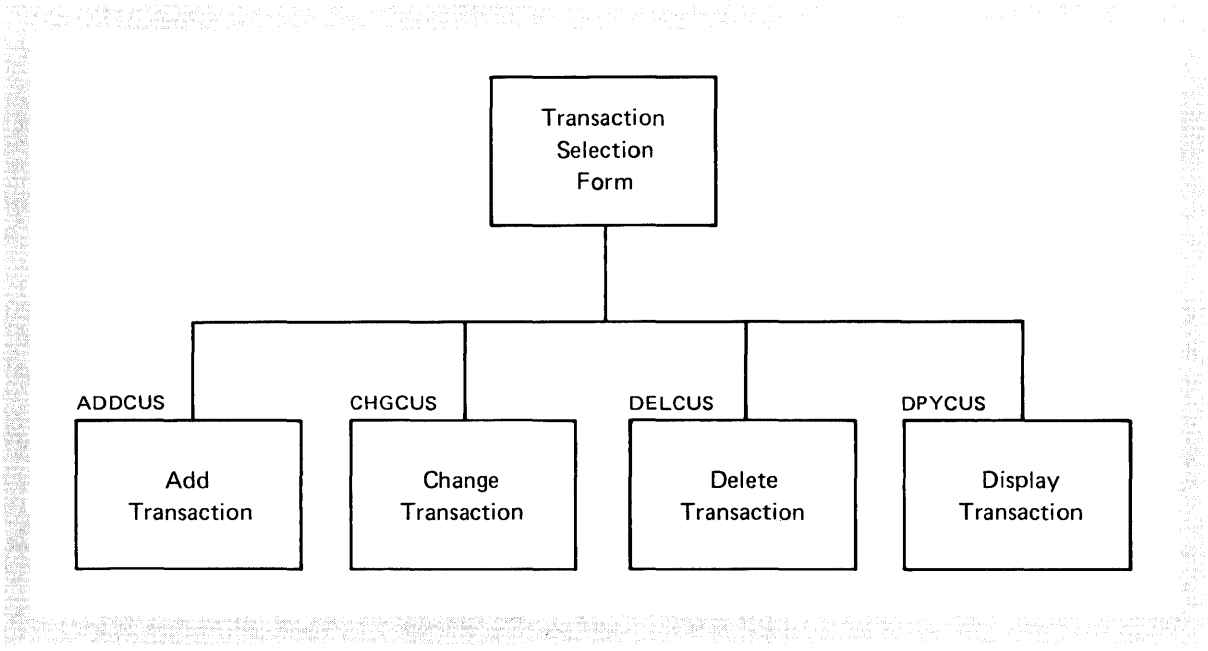


Figure 2-2 Structure of Sample Transaction Processor

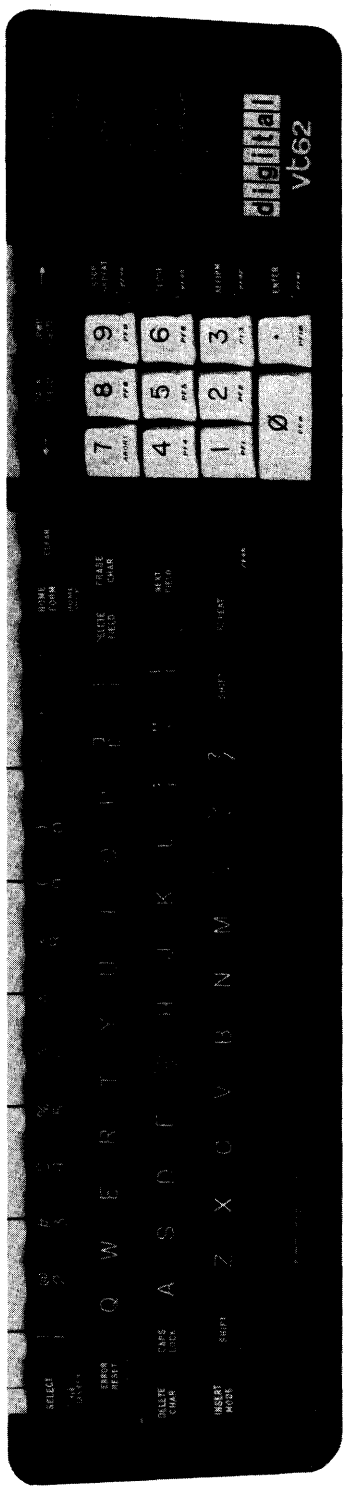


Figure 2-3 VT62 Video Terminal Keyboard

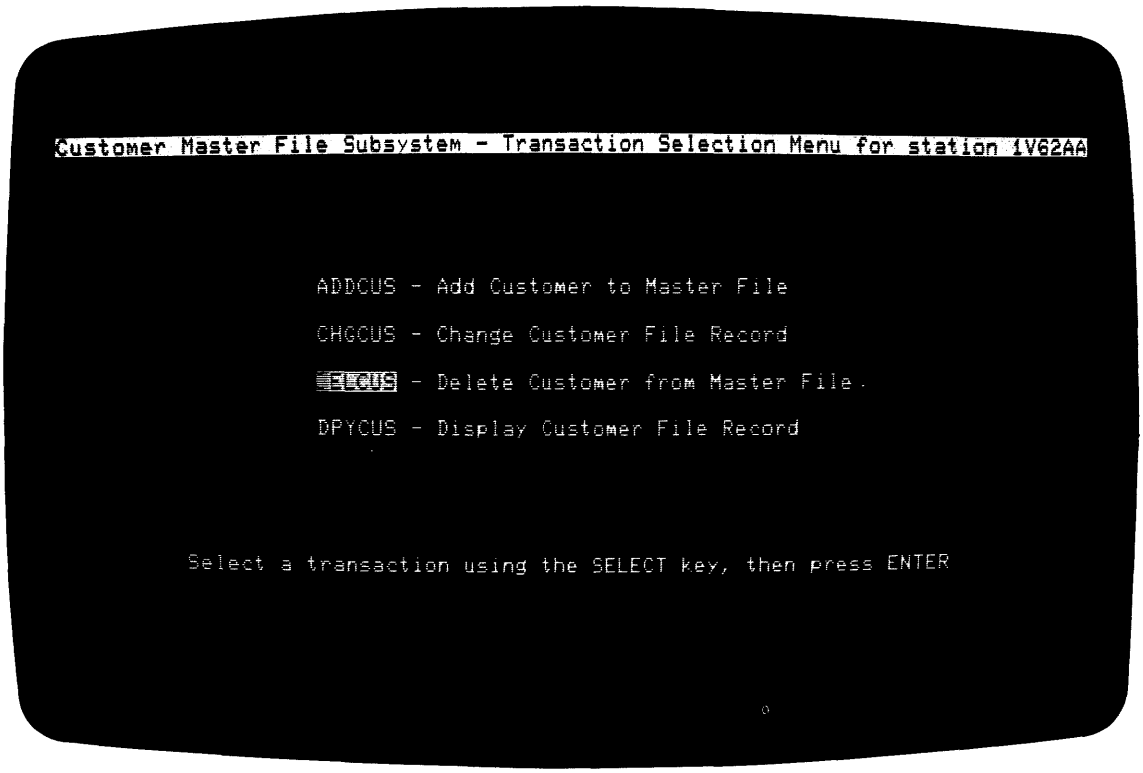


Figure 2-4 Delete Customer Transaction is Selected

If you make a mistake in selecting a transaction, you need only deselect the transaction by pressing the DESELECT key, which is the shift form of the SELECT key. After you press the DESELECT key, the DELCUS is displayed in a normal manner indicating that it is no longer "selected."

To demonstrate the operation of the sample application, you can add a record. To do this, first select the ADDCUS transaction by positioning the cursor on the letter A. To move the cursor back to this position, press the HOME DISP (home display) key. The only positions on this form which the cursor can be moved are the initial letters of ADDCUS, CHGCUS, DELCUS, and DPYCUS which are the names of the different transactions assigned by the application programmer. To select the ADDCUS transaction, press the SELECT key; in response, "ADDCUS" appears in reverse video. (See Figure 2-5.)

In this example, the six character fields containing the words ADDCUS, CHGCUS, DELCUS, and DYPCUS are known as Menu fields. Menu fields, which are features of the VT62, allow you to select one item from a group of items using the "select" procedure described above. Menu fields allow you to specify a transaction without having to remember the name of the transaction.

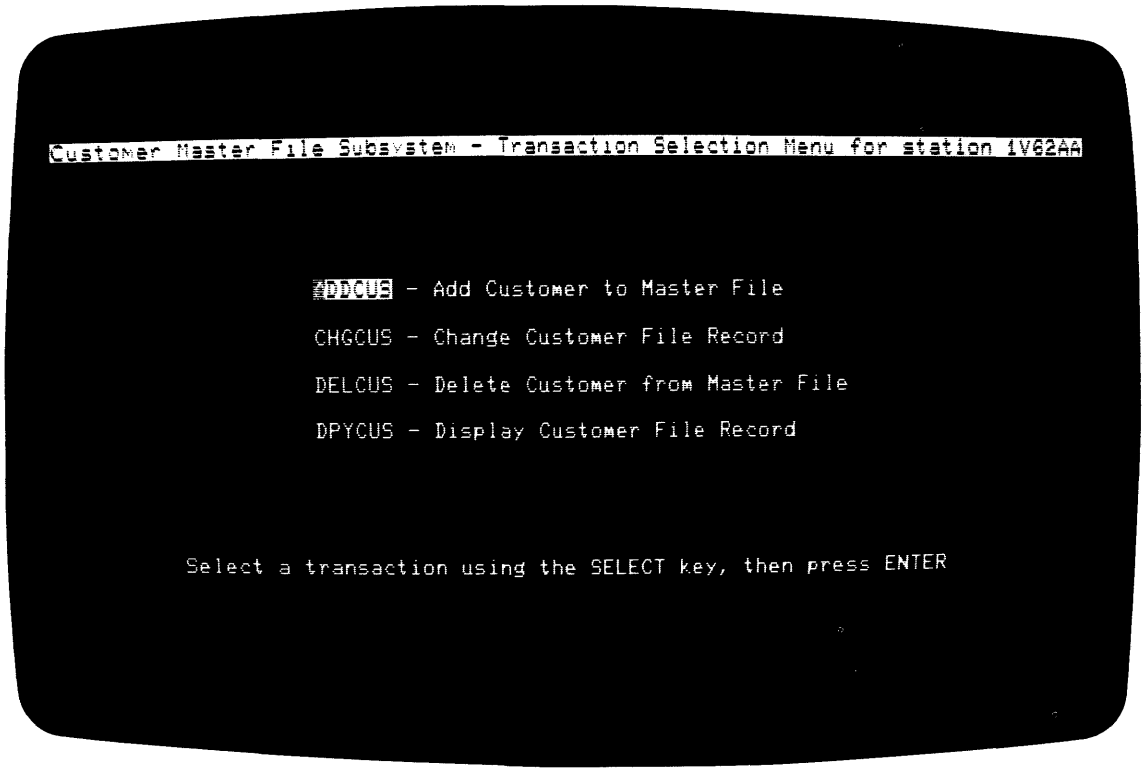


Figure 2-5 Add Customer Transaction is Selected

2.2 ADD TRANSACTION

After you select the transaction, press the ENTER key to display the first form in the add transaction (Figure 2-6). The form consists of a line on top that identifies the transaction, a line of instructions, a series of blank fields for customer data, and a final line that instructs you on what to do after you enter data. The customer number is shown as question marks to indicate that it is to be supplied by the program. In fact, in the example shown here, you cannot write into the customer number field; attempts to type characters in will not have any effect on the field.

To enter a customer record, first type the name and address of the customer. If the address consists of only two lines, the third line of the three shown on the screen may be omitted. Use the NEXT FIELD key in between lines entered. This key performs typewritter carriage return, moving the cursor from a field on one line to the beginning field on the next line. Next, you enter the telephone number. As one field is filled, the cursor automatically skips to the next field where data is required. This feature is known as automatic tabbing.

Figure 2-6 First Form of the Add Transaction

When typing the telephone number, you might accidentally type a letter rather than a number. Since TRAX is designed to accept only numeric characters in a numeric field, the keys lock, and a "NUMERIC ONLY" error message is displayed at the bottom of the screen (Figure 2-7). Notice that, the erroneous letter, is not displayed on the screen. In order to cancel the error message, press the ERROR RESET key and data entry can proceed normally. This error detection occurs completely within the terminal, the VT62 terminal initiates the audible alarm, locks the keyboard, and displays the error message. No system intervention is necessary.

Retype the telephone number, and then, if appropriate, type the "Company Contact" line. This is the line on the bottom of a mailing address in the form:

Attention: John Doe

The Company Contact line is often used when the customer name is a company. Finally, type the credit limit of the customer in question. The credit limit field is a right-justified field. That is, as each character is entered, the characters before it move to the left to make room for the new character, which always remains at the right margin of the field.

Figure 2-7 Terminal Error Message

As you enter the above data on the form shown in Figure 2-8, notice how the reverse video fields help in showing how many spaces remain in the field.

To enter this record in the master file, you need only press the ENTER key. If you press the CLOSE key instead, the record is not written and the Transaction Selection Form reappears. When you press the ENTER key, the system formats and sends the data from the screen to the application program, which then checks it and writes it to the customer file. After it writes the record to the customer file, the system modifies the form on the screen so that it appears as shown in Figure 2-9. Notice that the customer number is assigned by the program, the word "TRANSACTION COMPLETE" appear on the screen, and the message "Function Keys: Press AFFIRM to Add Another Customer – Press CLOSE to quit" appears on the bottom of the form. All of these features are determined by the application programmer.

At this point in the processing, only two function keys with one exception, mentioned below are "enabled" that is have any effect on the system: AFFIRM and CLOSE. Pressing the ENTER key produces no result. If you press the AFFIRM key, the initial blank form (Figure 2-6) reappears and you are ready to enter another customer record. However, if you want to display the record you just typed in order to verify that it is correct, press the CLOSE key. This causes the initial Transaction Selection Form to reappear so you can select the display transaction.

Figure 2-8 Completed Record for Customer

Figure 2-9 Customer Number Supplied by System

The third enabled function key is the ABORT key (Shift "7" on the numeric keypad to the right of the main keyboard). The ABORT key is always enabled, that is, always functions. It causes the system to abandon all processing and bring you back to the Transaction Selection Form. For example, if you type all the data shown above and then press the ABORT key, all of the data is lost.

When "staging" is employed, the ABORT key allows you to terminate an erroneous transaction without concern over what changes to the files were made as a result of the error. See the discussion of staging in Chapter 4 for more information. Although the ABORT key is always enabled, it is not referred to specifically in the remainder of the manual. Therefore the statement "Only the CLOSE and ENTER keys are enabled" actually means "Only the CLOSE, ENTER and ABORT keys are enabled."

2.3 DISPLAY TRANSACTION

Having pressed the CLOSE key, the Transaction Selection Form reappears. Press NEXT FIELD three times to move the cursor to "DPYCUS – Display Customer File Record" and then press SELECT. The word "DPYCUS" appears in reverse video. Press ENTER, and the first form of the display transaction appears, as shown in Figure 2-10. Notice that you can use either customer number or customer name as a way of specifying a unique customer.

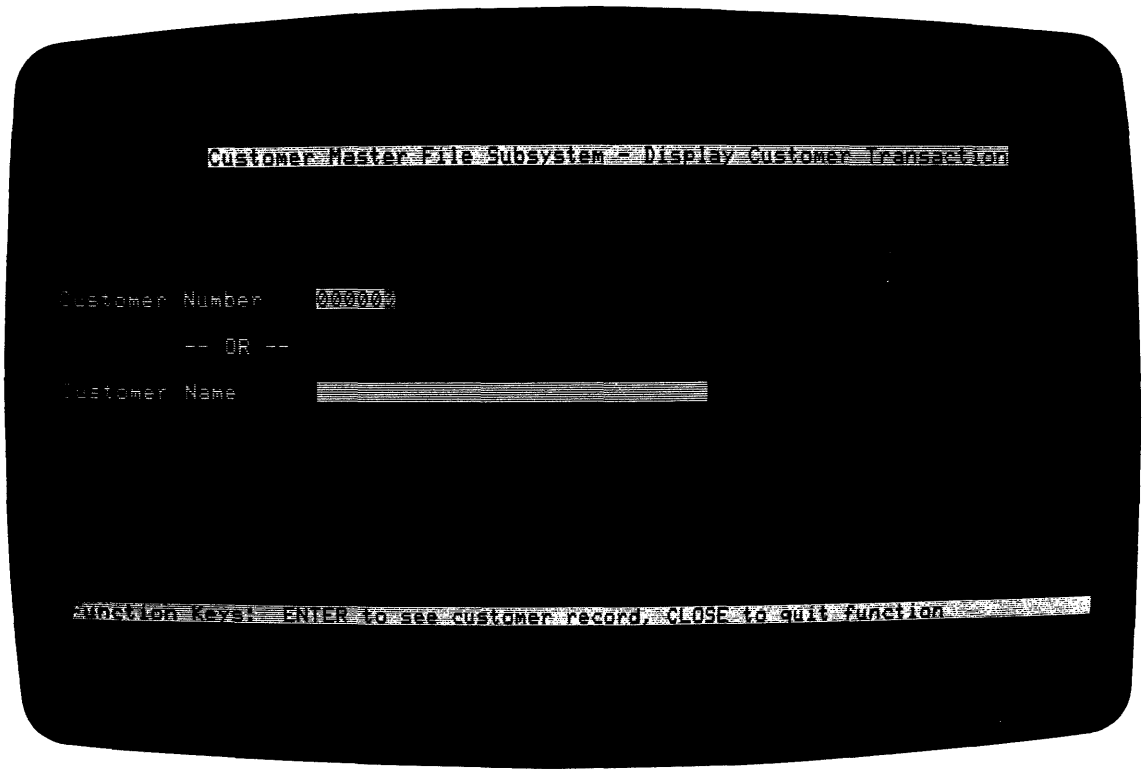


Figure 2-10 First Form of Display Transaction

If, for example, you decide to specify the record by name. You then press the NEXT FIELD key to move the cursor to the customer name field and type the name which you previously entered for the customer: "SMITH, ROBERT T". The name must, of course, be typed exactly as before. The form now appears as in Figure 2-11. Press the ENTER key, and the second form of this transaction (Figure 2-12) appears, showing the data previously entered. Notice that this form is different from the add transaction form on which the data was originally entered. The data fields on the right side of the form are new: Current Balance, Purchase to Date, Next Order No. and Next Payment No. These fields are all set to zero since your customer has not yet made any purchases. These new items represent additional data in the customer file which is accessed by other transactions. The current sample application is designed to be a subset of a larger application.

One other way in which the display transaction form differs from the add transaction form is that you cannot enter any data or move the cursor in the display transaction form.

Figure 2-11 Using Customer Name as a Key

Figure 2-12 Second Form of Display Transaction

After you have displayed the record, only three keys on the keyboard have any effect on the system (other than the ABORT key): ENTER, CLOSE and KEYDOT. The CLOSE key on the system, takes you back to the Transaction Selection Form. The "KEYDOT" key – the shift version of the (.) key on the numeric keypad to the right of the keyboard – can be used to print a copy of a given record on a hard copy terminal. The ENTER key allows you to browse through the file in alphabetic order by customer name. Press the ENTER key, and assuming there are other records in the file, a new record, such as shown in Figure 2-13, will appear on the terminal. You may continue pressing ENTER and a new record appears each time, arranged in alphabetic order by name.

In this last example, you displayed customer records based on customer name. If you want to display a record based on customer number first press CLOSE to get back to the Transaction Selection Form, again select the display transaction and press ENTER. This procedure causes the first form of the display transaction (Figure 2-10) to reappear. Since the customer number of the record you entered is 2001, as an experiment, try typing one less than that number, i.e., 2000 (Figure 2-14). Press ENTER, and the record shown in Figure 2-15 appears on the screen. Notice that the customer number is, in fact, 2000. Press ENTER for "next record". The record for Customer Number 2001, the same record displayed in Figure 2-12, appears on the screen. Press ENTER again, and the "Reached End-of-File" message reappears on the screen, as shown in Figure 2-16. The reason for this is that the record which you entered will have the highest customer number in the file (assuming that no one is entering data on another terminal at the same time). Therefore, an attempt to read the next (in numeric sequence) record in the file will fail.

Figure 2-13 The Next Record in the File

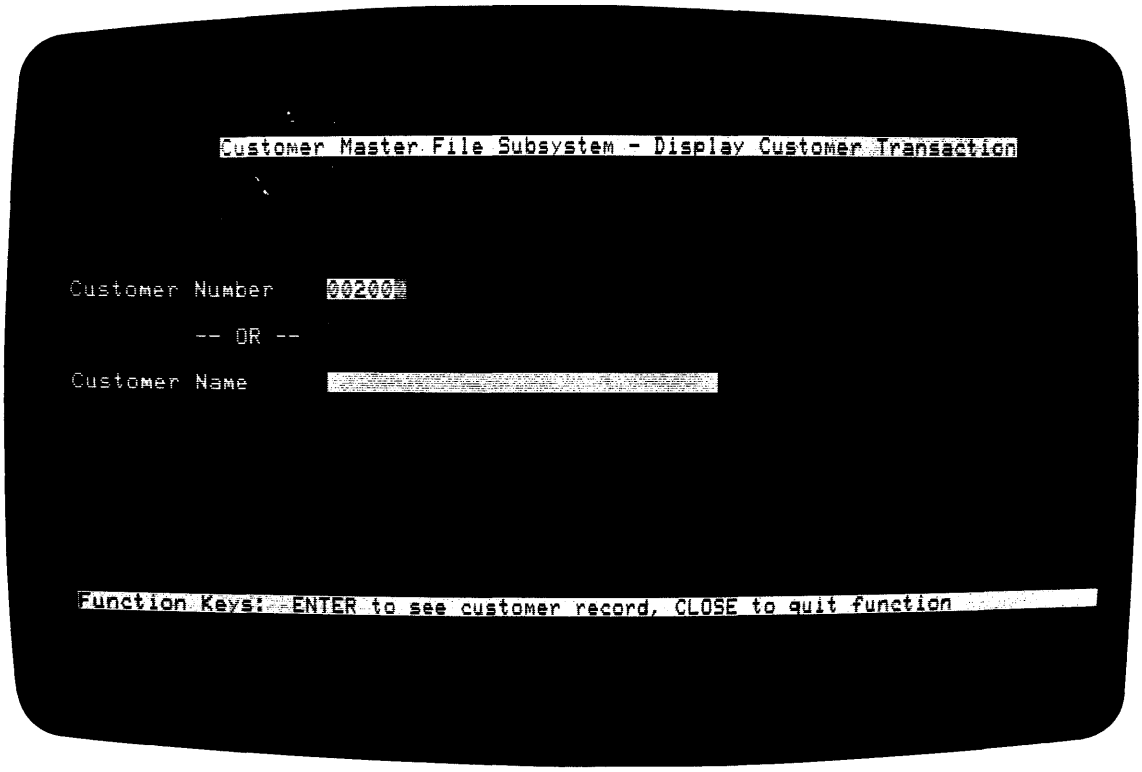


Figure 2-14 Using Customer Number as a Key

Figure 2-15 Displaying the Record for Customer 2000

Figure 2-16 Display Form Showing End-of-File Message

This error message is generated by the application program and not, as in the previous case, by the VT62 terminal. In addition to the "end-of-file" message, several codes are printed which would be useful to the application programmer in case of severe file errors.

2.4 CHANGE TRANSACTION

The change transaction is used to make changes to a customer record. To invoke the change transaction, first press the CLOSE key to exit the display transaction. The Transaction Selection Form appears. Select the change transaction (CHGCUS) and press ENTER. The first form of the change transaction then appears. For the change transaction, you must identify the individual record by customer number. Enter the customer number (in this case 2001) of the record you typed previously; the form will then appear as shown in Figure 2-17.

Press the ENTER key and the customer record appears as in Figure 2-18, just as previously entered. Notice that the form is essentially identical to that used for the add transaction, except for the header line which now reads "Change Customer Transaction", and the function keys line on the bottom of the screen which refers to "refiling" the record.

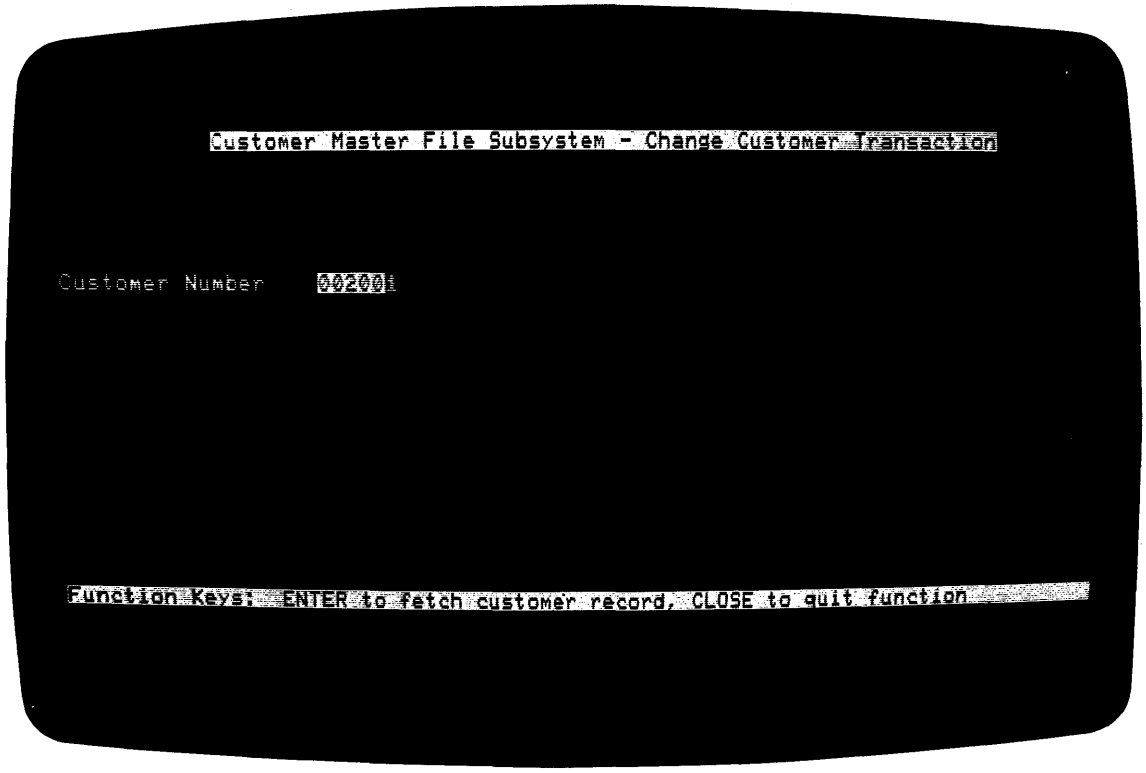


Figure 2-17 First Form of Change Transaction

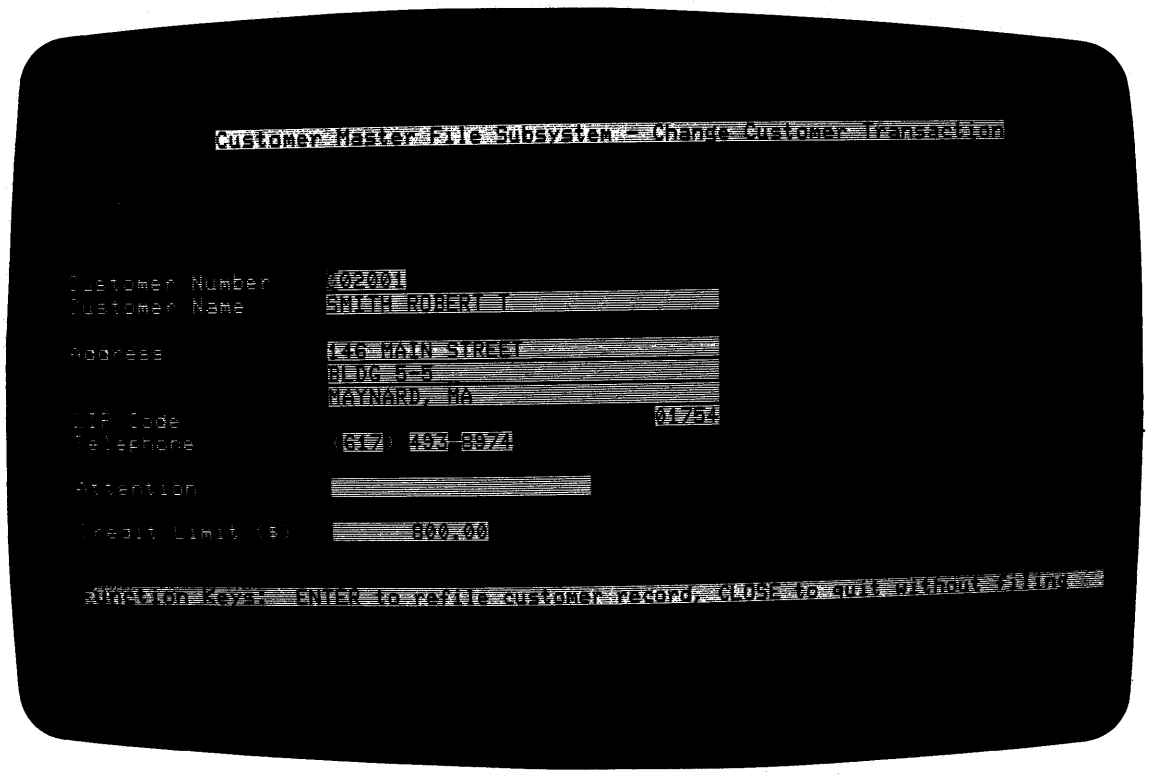


Figure 2-18 Second Form of Change Transaction

Suppose that you want to make a complete address change for the customer. To do this, first press the FORWARD FIELD key twice to position the cursor at the beginning of the address field. Retype each field and press NEXT FIELD. The retyped version is shown in Figure 2-19. After you type the correct address, press the ENTER key to refile the record. The program then prints "TRANSACTION COMPLETE" and "Function Keys: AFFIRM to proceed," as shown in Figure 2-20. At this point, if you press the AFFIRM key, the first form of the transaction, requesting customer number reappears, now you are ready to perform another change transaction. In order to get back to the Transaction Selection Form, press CLOSE.

2.5 DELETE TRANSACTION

To delete a record, first select the delete transaction (DELCUS) and press the ENTER key to display the first form of the delete transaction. (Figure 2-21) In order to delete the previously entered record, you type the customer number (2001) and press ENTER. Pressing ENTER causes the record to be displayed (Figure 2-22) so that you can verify that it is, the correct record. Notice that the form is similar to those in the add and change transactions.

Figure 2-19 The Address Change Has Been Typed

Figure 2-20 The Change Has Been Made to the File

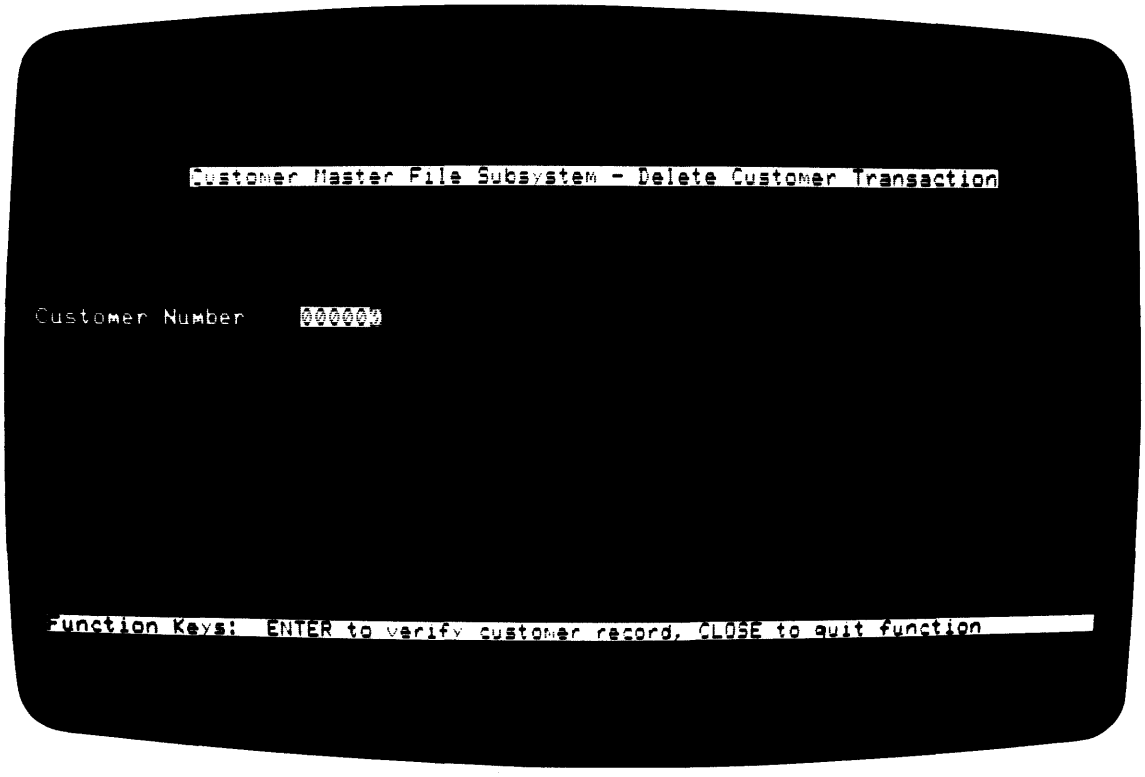


Figure 2-21 First Form of Delete Transaction

Figure 2-22 Second Form of Delete Transaction

Since the record is the correct one, press the ENTER key. Had the record been an incorrect one, pressing the CLOSE key would prevent it from being deleted, and would take you back to the Transaction Selection Form. Pressing ENTER causes the message "TRANSACTION COMPLETE" and the instructions "AFFIRM to proceed" to be displayed, as shown in Figure 2-23. Pressing AFFIRM then takes you back to the first form (Figure 2-21) of the delete transaction, ready to delete another record.

Figure 2-23 The Record Has Been Deleted

Again it should be emphasized that the sample application described here is only an example of a small application written by one application programmer. Although it demonstrates a number of features of the TRAX system, it represents only one of a large number of possible ways of coding this application.

CHAPTER 3

TRAX SYSTEM OPERATION

This chapter describes the general operation of the TRAX Transaction Processing System, the components of a transaction processor, and the relationship of the components. Chapter 2 demonstrated the various transactions of a sample application. In addition, this chapter shows the relationship between components of the sample application developed in Chapter 2, and how you can create a transaction processor for a given application.

3.1 BASIC SYSTEM CONCEPTS

TRAX can support up to two transaction processors operating simultaneously (Figure 3-1). Each transaction processor consists of the TRAX Transaction Processing Executive including Data Management Services (RMS), the application terminals where transactions are performed, and the three components; forms definitions, transaction step tasks, and definition files which the user defines for the application.

3.1.1 Forms and Forms Definitions

A form is the structured arrangement of fields displayed on a video terminal. The word form may also apply to structured data printed on a hard-copy device such as the LA-180 printer. In this case, it is called a report form. Forms are the medium through which the application program and the terminal user communicate.

The use of forms in conjunction with the VT62 terminal has distinct advantages over conventional data entry techniques, such as the DISPLAY and ACCEPT statements in COBOL, or the PRINT and INPUT statements in BASIC. Some of these advantages are:

- Forms allow the data comprising an entire screen to be transmitted at one time. Since all preliminary work is handled by the VT62 terminal, system overhead is minimized, making more computer time available to speed up transaction response time.
- Forms allow the user to see a complete screen of data, thereby clarifying the interrelationship between data items for the user. Also, changes can be made in one operation, and the results visually verified before being entered.
- Forms, takes the place of conventional paper forms, so that the user deals with data in a familiar manner.

You code forms using the TRAX Application Terminal Language (ATL) which is a series of simple English-like statements which define the nature of every field on the screen. These same statements also predefine possible modifications of those fields such as displaying responses for the user.

The programmer uses DEC EDITOR to create a file of statements in ATL source language (see Figure 3-2) and then compiles this file using the ATL utility program. The result is a record in the Forms Definition File which contains information to instruct the TRAX executive on how to display a given form on the VT62 terminal.

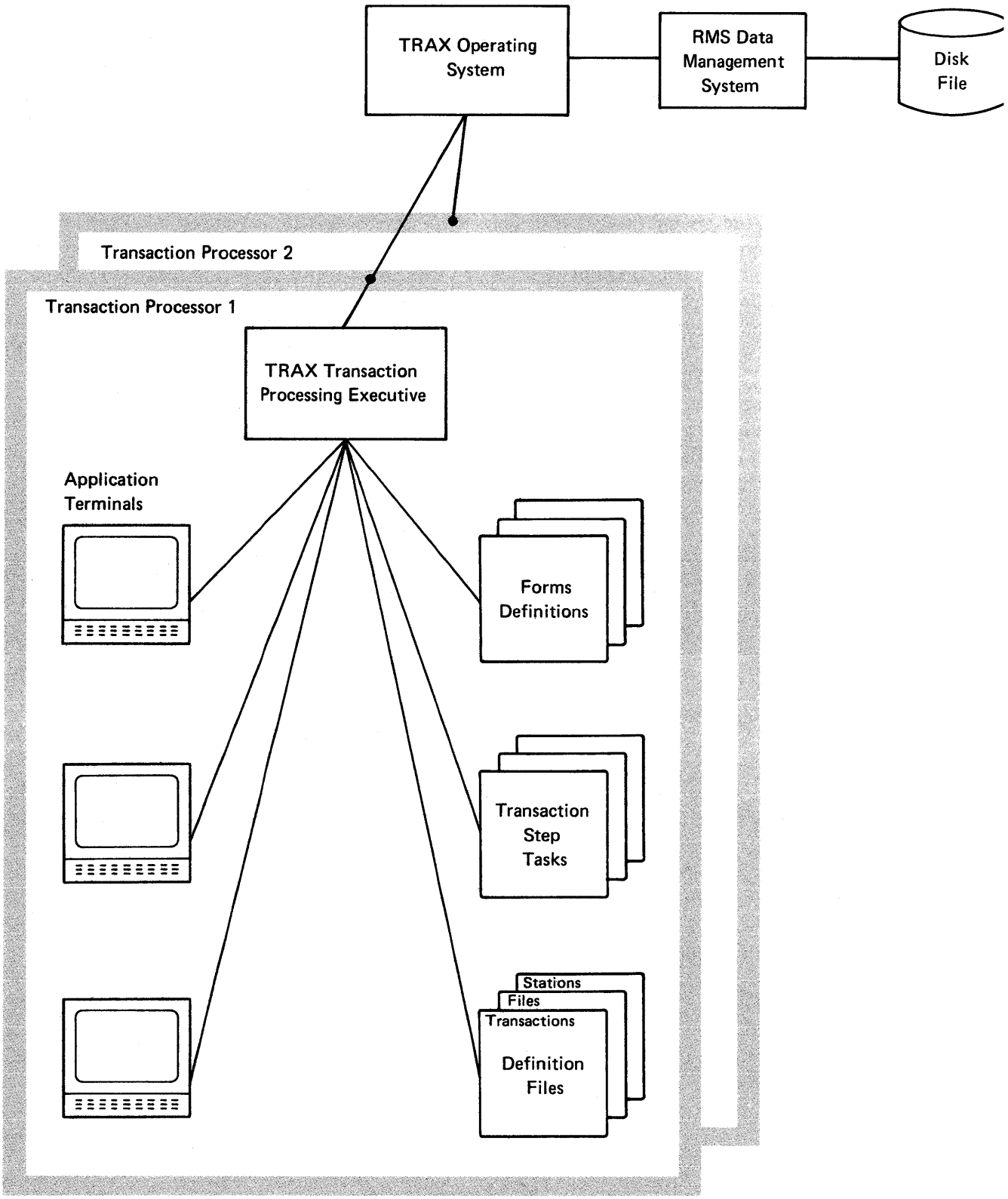


Figure 3-1 Transaction Processor Structure

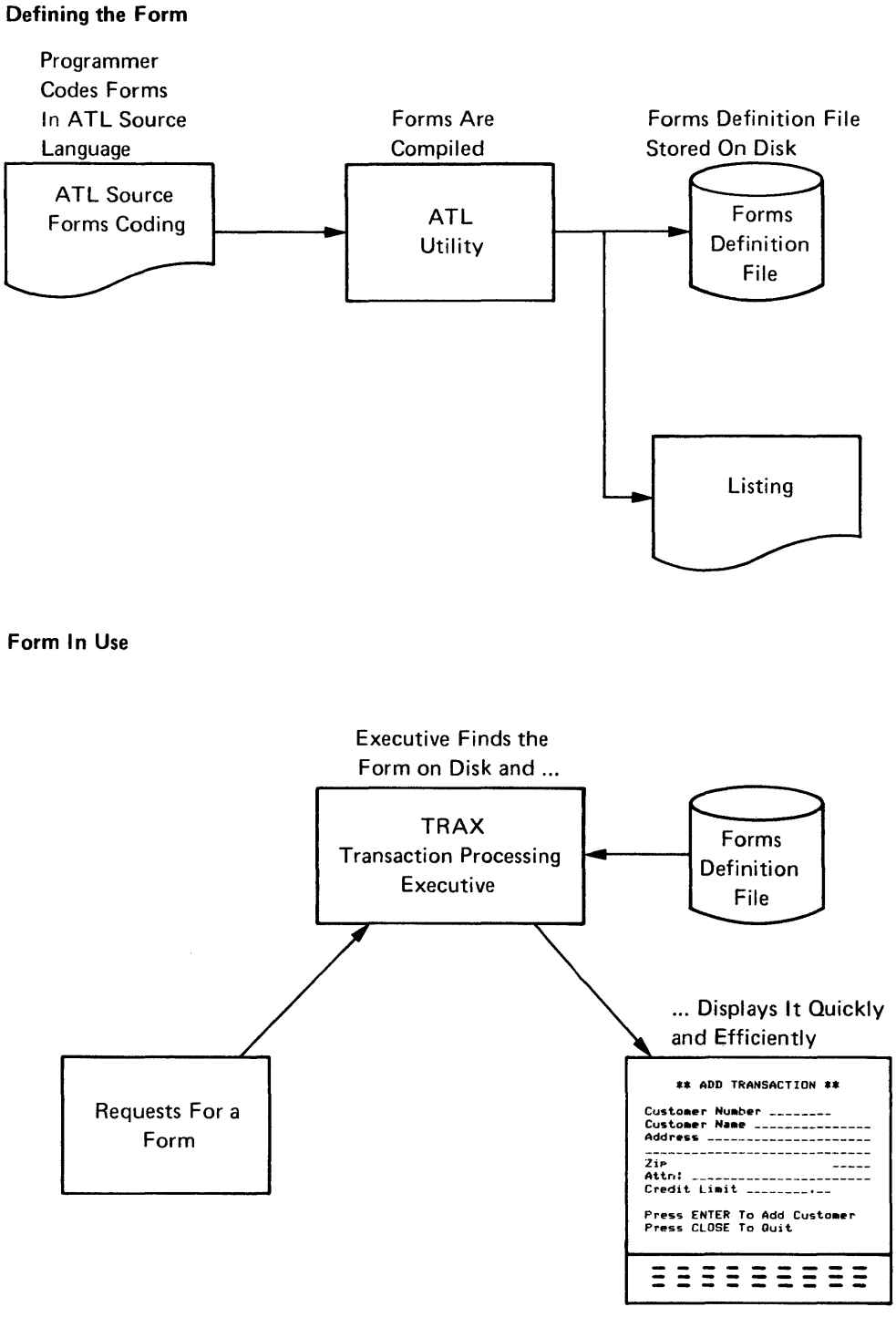


Figure 3-2 Defining and Using Forms

In addition to defining the data fields appearing on the displayed form, the form definition record contains codes that predefine modifications to the form such as error messages, to inform the user of processing errors, and user instructions concerning error recovery. Such modifications change certain fields of the form, but do not alter the basic form identity.

Appendix B contains a sample of an ATL source file. Further explanation is provided in the *TRAX Application Terminal Language (ATL) Reference Manual.*

3.1.2 Transaction Step Tasks (TSTs)

Transaction Step Tasks (TSTs) are user-written application programs written in the form of a subroutine. Each TST performs one processing step of the pertinent Transaction.

In this manner TRAX allows the application programmer to segment the processing of exchange data into a series of small sequentially executed application programs.

The Transaction Processing Executive controls the simultaneous execution of many TSTs, a set of which performs the processing for each transaction.

Figure 3-3 shows the data flow between a form and two TSTs for the add transaction described in Chapter 2. The symbol in the upper left corner of Figure 3-3 that resembles a video terminal is used to represent the form that appears on the screen. The user types in customer data on the form and presses the ENTER key. The system then transmits this data to TST 1 which validates it. Then the same data is passed on to TST 2, which writes it into the customer file on disk. Finally, TST 2 sends a message back to the form to tell the user that the transaction is complete, and to instruct the user as to the next step to take. The second illustration of the form is shown as dotted to indicate that this is not a new form, but a modification to the original form. Coding for a sample TST, in COBOL, is shown in Appendix B.

3.1.3 Transaction Instances

When discussing transaction processing, it is necessary to distinguish between a transaction in general, (that is, the definition of the transaction), and a specific example of a transaction running on a terminal. A transaction running on a terminal is known as a transaction instance. The word instance in this context is used in its normal dictionary meaning, which is "a case or example."

Figure 3-4 illustrates a transaction processor with four terminals and distinguishes between transaction and transaction instance. Three of the terminals are running an add transaction and one is running a change transaction. The illustration, then, shows two kinds of transactions and four transaction instances altogether. Although the three add transactions represent the same transaction definition, each is a unique transaction instance. The data entered by the user and processed by the system for each transaction instance is totally different. Each transaction instance can proceed at its own rate, depending on the available system resources and the speed with which the user enters data.

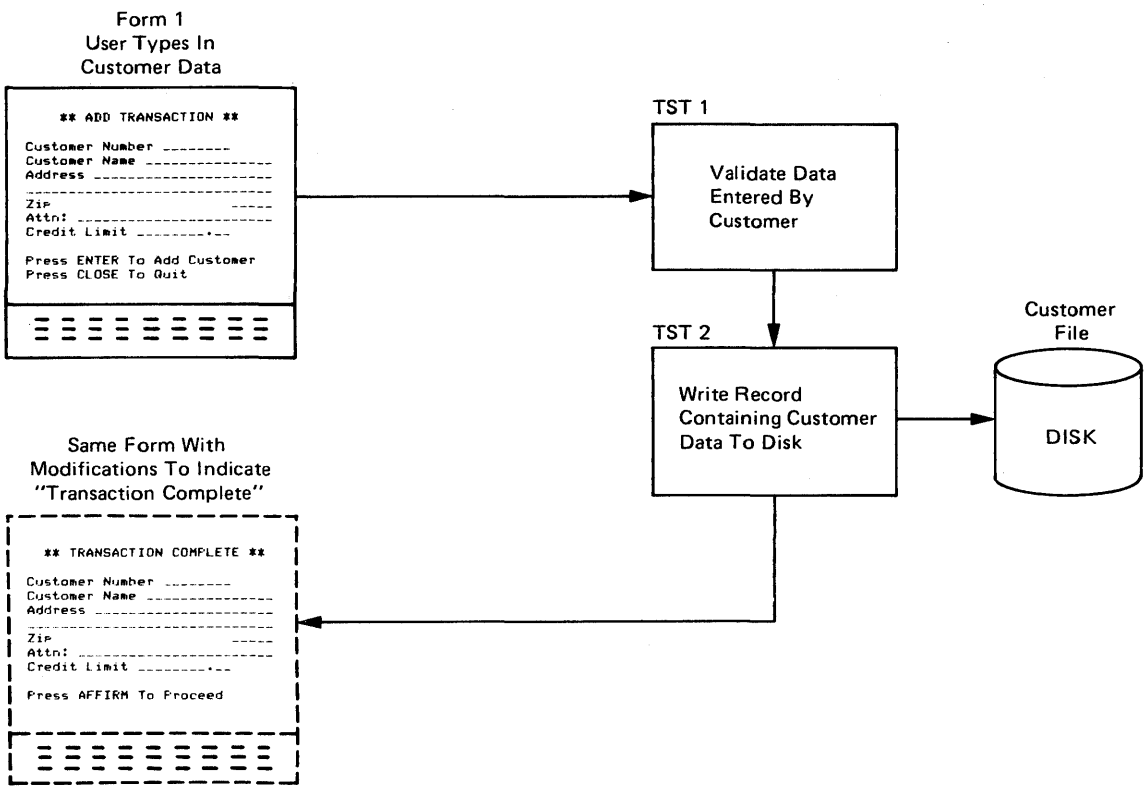


Figure 3-3 An "Add" Transaction Showing a Form and Two TSTs

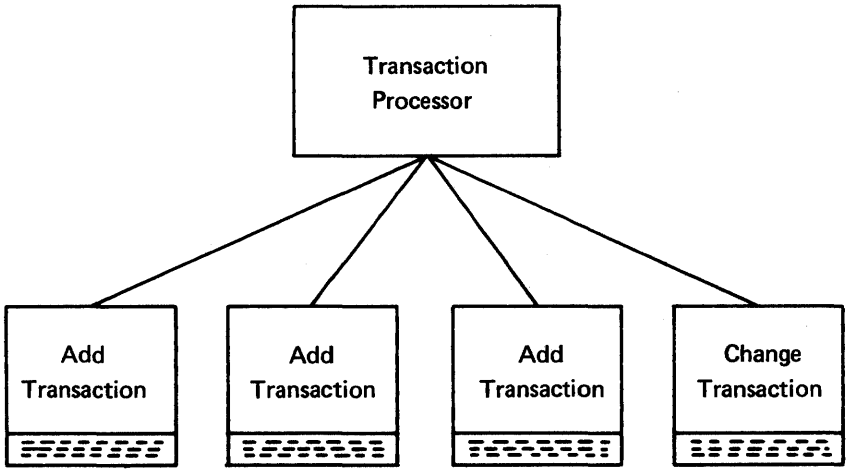


Figure 3-4 Four Transaction Instances

3.1.4 Exchange Messages

In the sample transactions described in Chapter 2, a common data entry sequence involved typing some data and then pressing the ENTER key. Upon receipt of the data, TRAX formats the data into an exchange message according to the forms definition. An exchange message is a formatted sequence of data sent from a form to one or more TSTs. Figure 3-5 illustrates this operation for the add transaction shown in Chapter 2.

After TRAX formats the data, it sends it to the series of TSTs that will process it. This is illustrated in Figure 3-6. The exchange message goes to the first TST, where the data in the exchange message is processed.

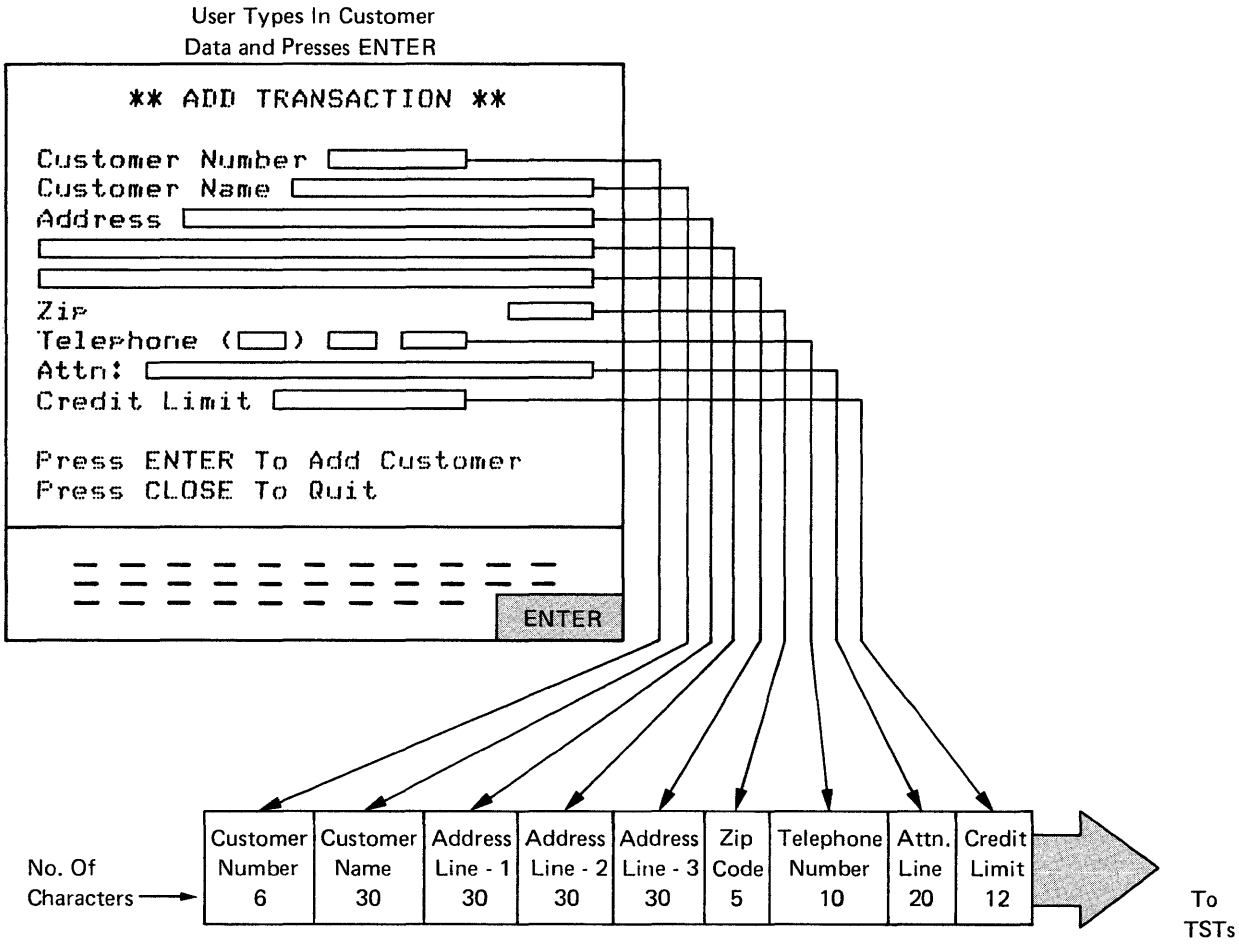


Figure 3-5 Creation of a Typical Exchange Message

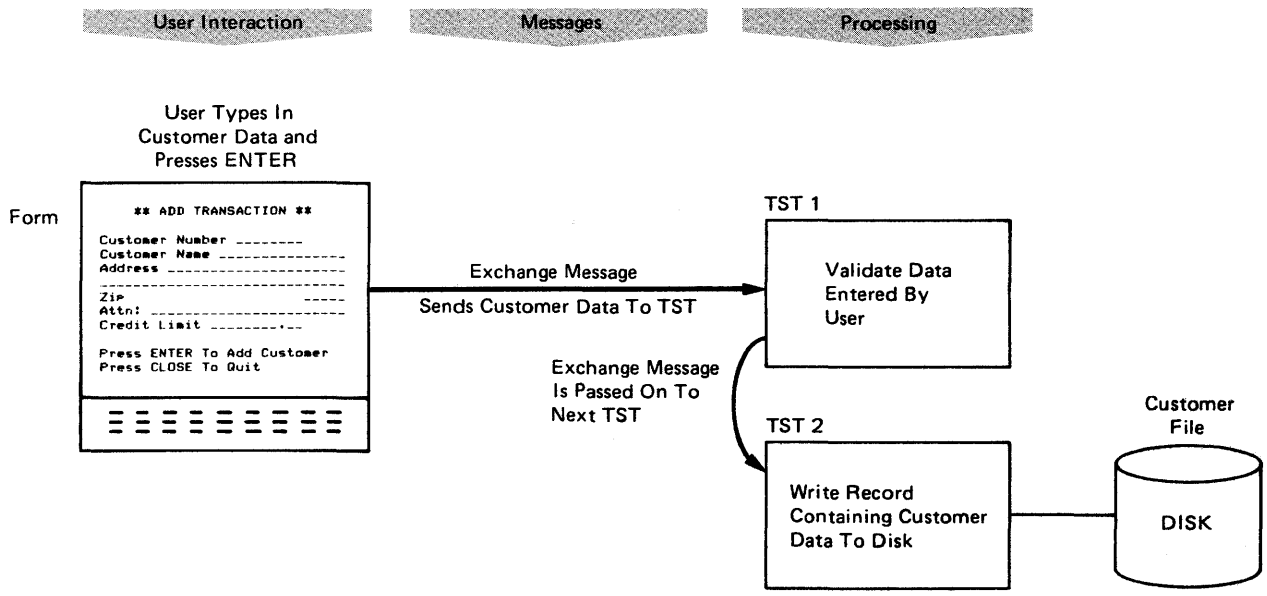


Figure 3-6 Transfer of the Exchange Message

A routing list associated with each exchange message specifies the sequence of TSTs that will process the particular exchange message. Notice that the diagram shown in Figure 3-6 is divided into three sections: user interaction, messages, and processing. The user interaction section comprises forms and user interaction with the forms. The message section shows the messages that are sent between forms and TSTs. The processing section consists of the TSTs that perform the actual processing for the application.

3.1.5 Response Messages

The data structure generated by a TST and directed to the user's terminal is called a response message (Figure 3-7). The response message is the means by which TSTs return data to the user's terminal. When a terminal interface, or other data source, sends an exchange message to be processed, it will wait for only one response message. Therefore, a single TST cannot send multiple response messages and only one TST of all those visited by an exchange message can send a response message.

3.1.6 Stations

A station is a message-handling location within the transaction processor. Stations can send and receive messages. There are several kinds of stations which are classified according to the kind of transaction processing element that uses the station to send and receive messages. The two most important kinds of stations are terminal stations and TST stations.

Each terminal station is associated with a particular application terminal. Data received from a terminal is formatted into an exchange message by the terminal station and sent to the appropriate list of processing locations. The response message for that exchange message will be received at the same terminal station, interpreted according to the current state of the user's conversation, and then forwarded to the appropriate terminal.

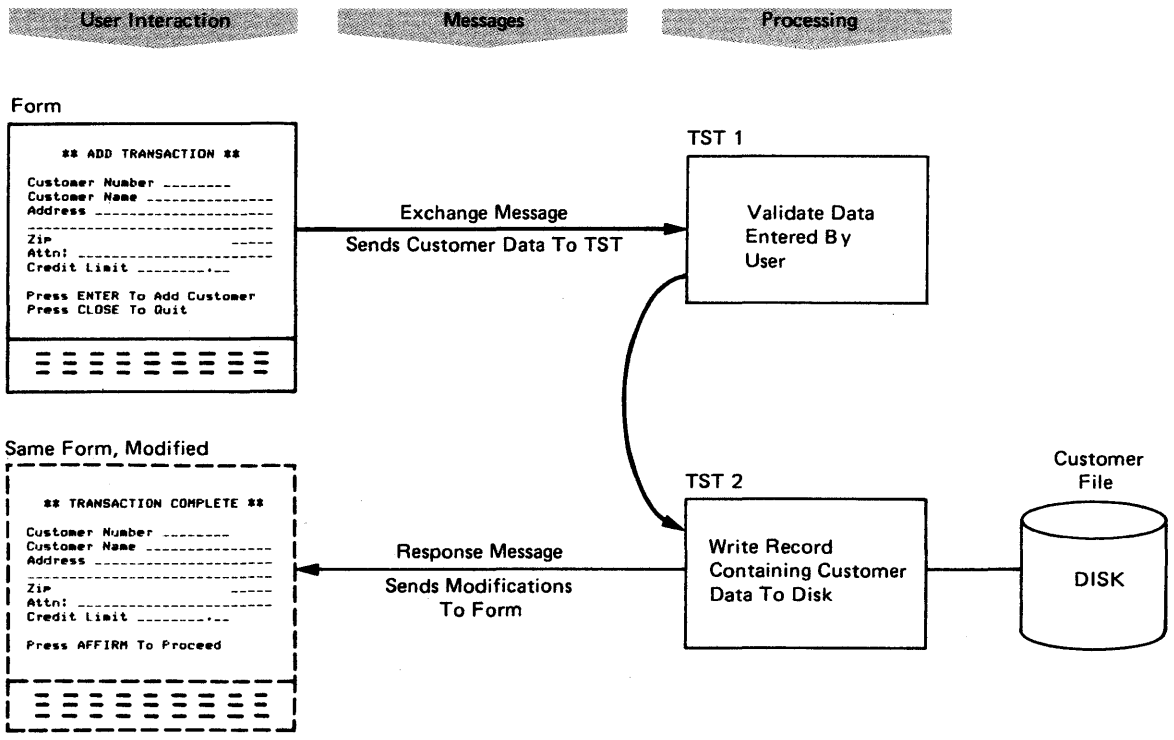


Figure 3-7 Transfer of the Response Message

TST stations are associated with Transaction Step Tasks (TSTs). Exchange messages received at a TST station cause a copy of the TST to be activated in order to process the exchange message. The number of copies of a TST can be limited and exchange messages in excess of this limit will await a free TST copy to process them.

3.1.7 Exchanges

An exchange is the basic unit of processing in the TRAX system. From the terminal user's view point, an exchange typically consists of three steps:

1. The system displays a form containing a request for data
2. The user types in data, and press the enter key
3. The system responds.

From an application programmer's view point, an exchange consists of an exchange message being routed to a series of TSTs. An exchange normally consists of the following elements in this order:

1. A form
2. An exchange message (generated when the user enters data and presses ENTER)
3. A series of one or more TSTs that process the exchange message
4. A response message, sent by one of the TSTs in the exchange.

A typical exchange is one shown in Figure 3-8. This is the same transaction used in the previous illustrations. Notice that this is a one-exchange transaction. However, a transaction can have many exchanges.

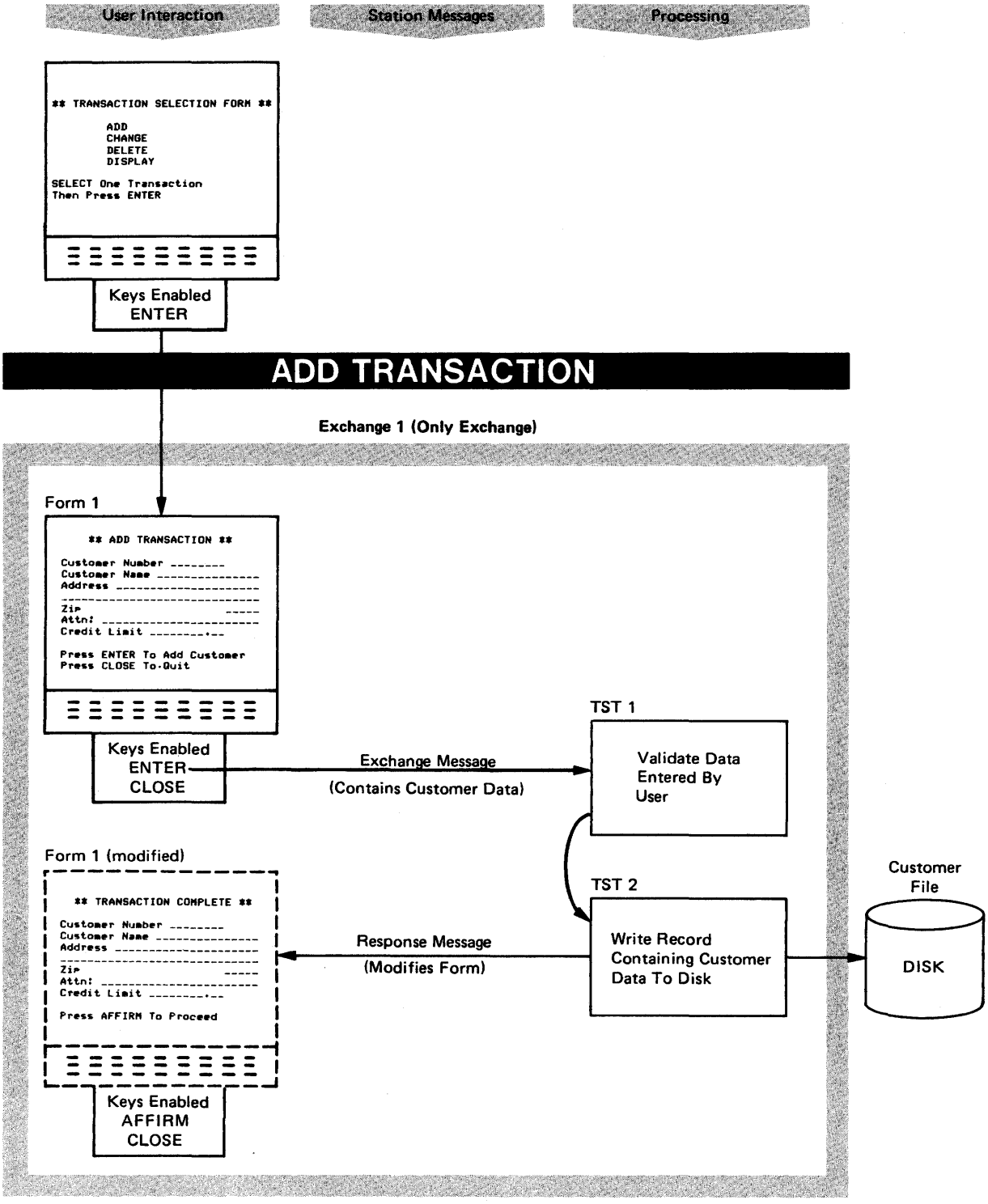


Figure 3-8 A One Exchange Transaction

In the add transaction shown in Figure 3-8, the user selects the add transaction and presses ENTER. In response, the system starts the transaction beginning with Form 1. The transaction selection process is not considered to be an exchange, since the transaction has not actually begun.

Figure 3-8 shows several new elements not shown in Figure 2-6: the Transaction Selection Form and a bar below indicating that the add transaction was selected. The exchange itself is enclosed in a gray box.

In addition, the symbol for a form contains a new box entitled "Keys Enabled." This box specifies which function keys such as CLOSE, AFFIRM, STOP-REPEAT, ENTER are permitted to function at this stage of the processing. If a key is disabled, pressing it has no effect. If a key is enabled, pressing it causes the pertinent function to be performed. The application programmer specifies which keys are to be enabled and which disabled when he codes the form for the exchange. A response message which modifies a form can alter the keys enabled.

In Figure 3-8, the exchange begins with a form which requests data from the user. The user types in the data and presses ENTER. The system then generates an exchange message containing this data and sends it to a preassigned routing list of TSTs. After TST 1 has validated the data, the exchange message passes to the second TST. TST 2 writes the customer data from the exchange message to disk. When it is finished, it sends a response message to the form changing the keys enabled, saying "TRANSACTION COMPLETE," and giving the user instructions on the next step to take.

It is common for a form to be changed during the processing that occurs in a given exchange. However, an exchange cannot have more than one form. The form may be modified – keys may be enabled or disabled, certain fields may be changed – but the basic form remains the same. For example, input fields, the fields into which the user types data, cannot be changed. Further, all changes must be predefined when the form is first coded.

It is evident that the structure of exchanges has a profound effect on the way in which an application designer structures a given application. For a given transaction, each exchange represents an "exchange" of information between the user and the system. The basic design process is essentially one of breaking the application down into appropriate transactions and exchanges.

3.1.8 User Function Keys

The VT62 keyboard (see Figure 3-9) has several keys known as user function keys. The user function keys are ENTER and the shift versions of 0, 1, 2, 3 and . (dot) located to the right of the keyboard on the numeric keypad (Figure 3-9). All of these keys have the same function as ENTER: each causes the system to generate an exchange message. However, by a simple step in coding a form, the keys can be made program distinguishable. Thus, the user function keys may be assigned a fixed function in a user's application program. For example, in a sales-statistics application, keys might have the followings meanings in regard to a specific report:

Key 0 – results, overall summary
Key 1 – results, summary compared with previous years
Key 2 – results, broken down by product line
Key 3 – results, broken down by territory

In this manner, the application designer may assign a logical set of functions to the keys.

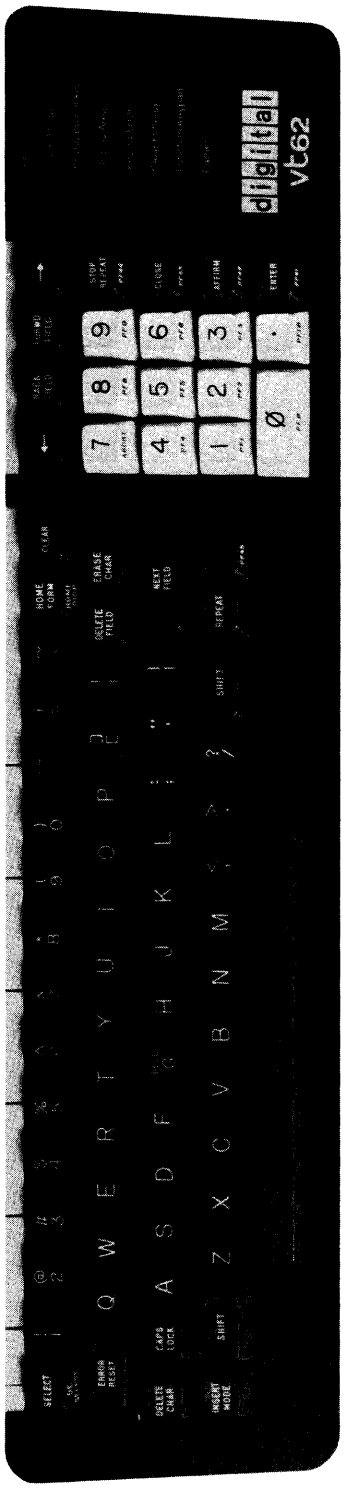


Figure 3-9 VT62 Video Terminal Keyboard

User function keys are made TST distinguishable through a TRAX function called key caps. A key cap is a string of text assigned to a given user function key when the application programmer codes the form. When a key is pressed, an exchange message is created which contains, in addition to other data, the text string assigned to the particular key pressed. These text strings can be a letter, a word or a phrase. For example, in a warehouse inventory application, certain keys might be assigned the following text strings:

Key . – ITEM MISSING FROM STOCK
Key 0 – ITEM PICKED (FULL COUNT)
Key 1 – ITEM PICKED (PARTIAL COUNT)
Key 2 – ITEM SUBSTITUTED

When the terminal user presses one of the function keys, the system sends an exchange message containing one of the text strings along with other data entered on the terminal. The application program then tests to see which text string was received, and performs the processing associated with that string.

3.1.9 System Function Keys

In addition to user function keys, the VT62 has several keys known as system function keys. System function keys are: AFFIRM, CLOSE, STOP REPEAT, and ABORT. System function keys do not cause an exchange message to be created; rather, allow the terminal user to specify one of several alternative paths in the system.

In the example shown in Figure 3-10, when the user presses AFFIRM at the end of the exchange, the exchange is restarted with a fresh copy of the form. Similarily, when the user presses CLOSE, the terminal returns to the Transaction Selection Form. After pressing a system function key and until a system response is displayed the keyboard is locked and no user input is possible.

3.1.10 Reply and Proceed Messages

Up to this point, all messages which originate from a TST were called response messages. There are, however, a number of different types of response messages. Two of the most important are reply and proceed messages. Other types are discussed in the *TRAX Application Programmers Reference Manual.*

The response message, shown in Figure 3-10, is an example of a reply message. A TST sends a reply message to a form in order to modify the form and to enable user input. In this example, the reply message: (1) changes the top of the screen to say "TRANSACTION COMPLETE," (2) changes the instructions at the bottom of the screen to read "Function Keys: Press AFFIRM to Proceed; Press CLOSE to quit function," (3) disables the ENTER key, and (4) enables the AFFIRM key. All of these actions were precoded on the form as reply number 2. When the TST sends reply message number 2 to the form, all of the above actions occur. The application programmer implements a reply message by calling a standard system-supplied subroutine called REPLY.

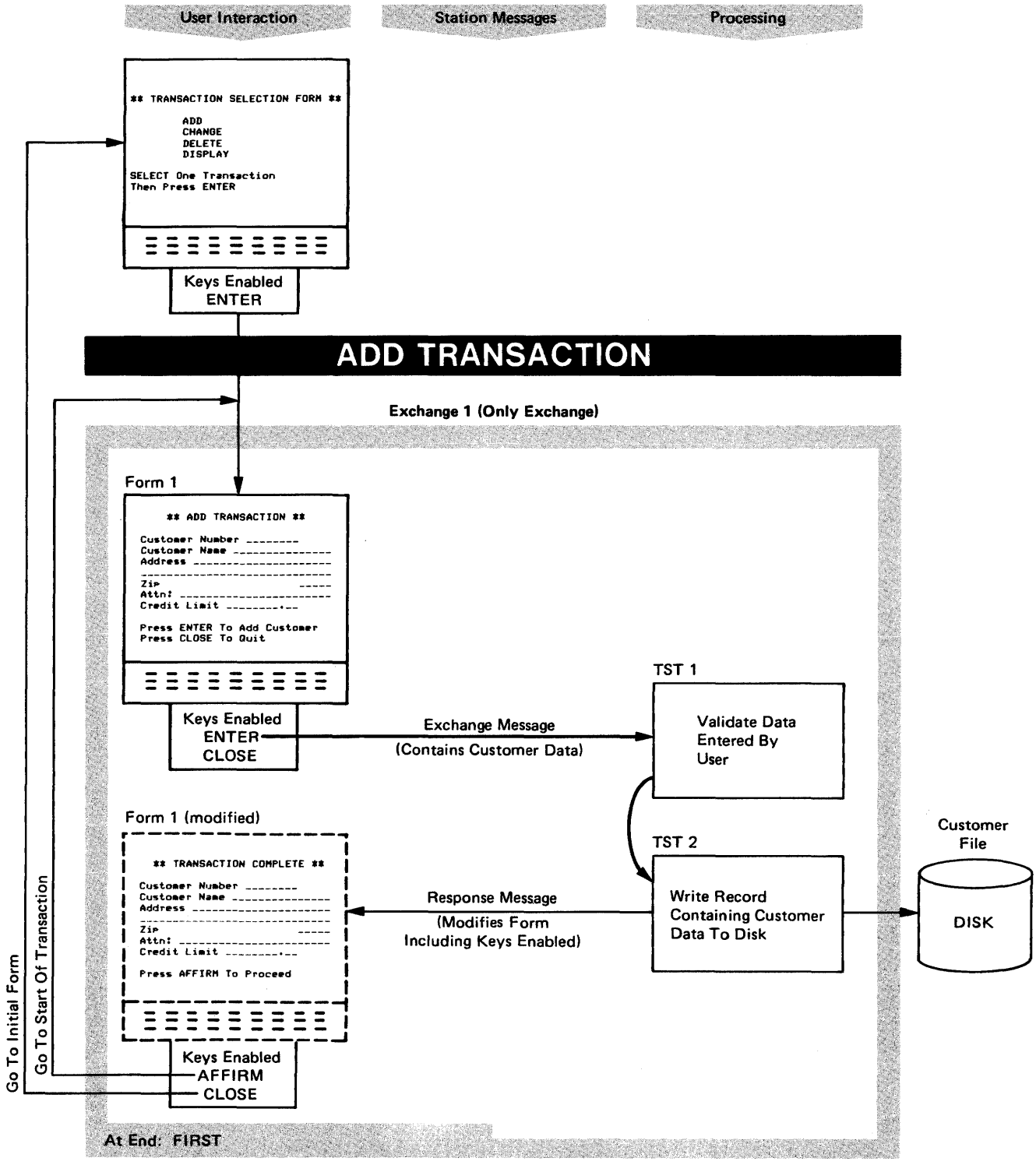


Figure 3-10 Example of System Function Key Actions

In addition to the specific functions involved in displaying the form (Figure 3-10), the system performs a general function of unlocking the keyboard. After the user presses ENTER, as shown in the top form of the diagram, the system locks the keyboard. The reply message unlocks the keyboard, although in this case it enables only the AFFIRM and CLOSE keys. One function of a reply message is to return control to the user. However, the form that the user sees when control is returned is the same form as before. A reply message can modify a form, but cannot create a new form.

A reply message always functions within the current exchange. If the designer wants to have a fresh version of the form at this point, it is necessary to leave the exchange. One way to do this is with the proceed message.

A proceed message is a message sent by a TST that causes the transaction to go to the exchange specified by the subsequent action. A proceed message can also optionally pass data, for example a record, to the new exchange. Figure 3-11, which is a diagram of the display transaction described in Chapter 2, illustrates this point. Notice that this is a two-exchange transaction.

3.2 HOW TO CREATE A TRANSACTION PROCESSOR

This section describes how the various components of a transaction processor interrelate, along with the TRAX system logic, and how you combine all components to create a transaction processor. The sample application described in Chapter 2 continues to be the basis for the following discussion. The intention in this section is to define the logical units in a TP and how you combine them as to create a transaction processor.

3.2.1 Structure of a Transaction Processor

There is a major difference between the structure of a TRAX Transaction Processor (TP) and that of an application program in a more conventional system. A conventional application program consists of a large code sequence, generally divided into subroutines or other logical units. This code, sequence along with the execution sequence and conditions, is contained in the body of the application program.

Compared to conventionally coded applications the file-defined logic of a TRAX TP makes it easier to understand and diagram application program flow. The basic program flow is documented by listing one file. This results in faster application design, faster coding, easier debugging, simpler maintenance, and greater ease in changing or upgrading an application program.

3.2.2 Organization of the Elements

The basic elements of a transaction are forms and TSTs combined into exchanges. To create a transaction processor, you must reduce the processing into transactions and then into exchanges, and then code the forms and TSTs for each exchange. You must then inform TRAX of the association between the pertinent forms and the TSTs which comprise the individual exchanges through use of the Transaction Definition Utility (TRADEF, see Section 3.2.3).

3.2.3 The "TRADEF" Definition File

The most important file for controlling the operation of a TP is the TRADEF (TRAnsaction DEFinition) file. The TRADEF utility creates this file through an interactive dialog with the application programmer. The TRADEF file contains the essential logic, in coded form, for each exchange in a transaction.

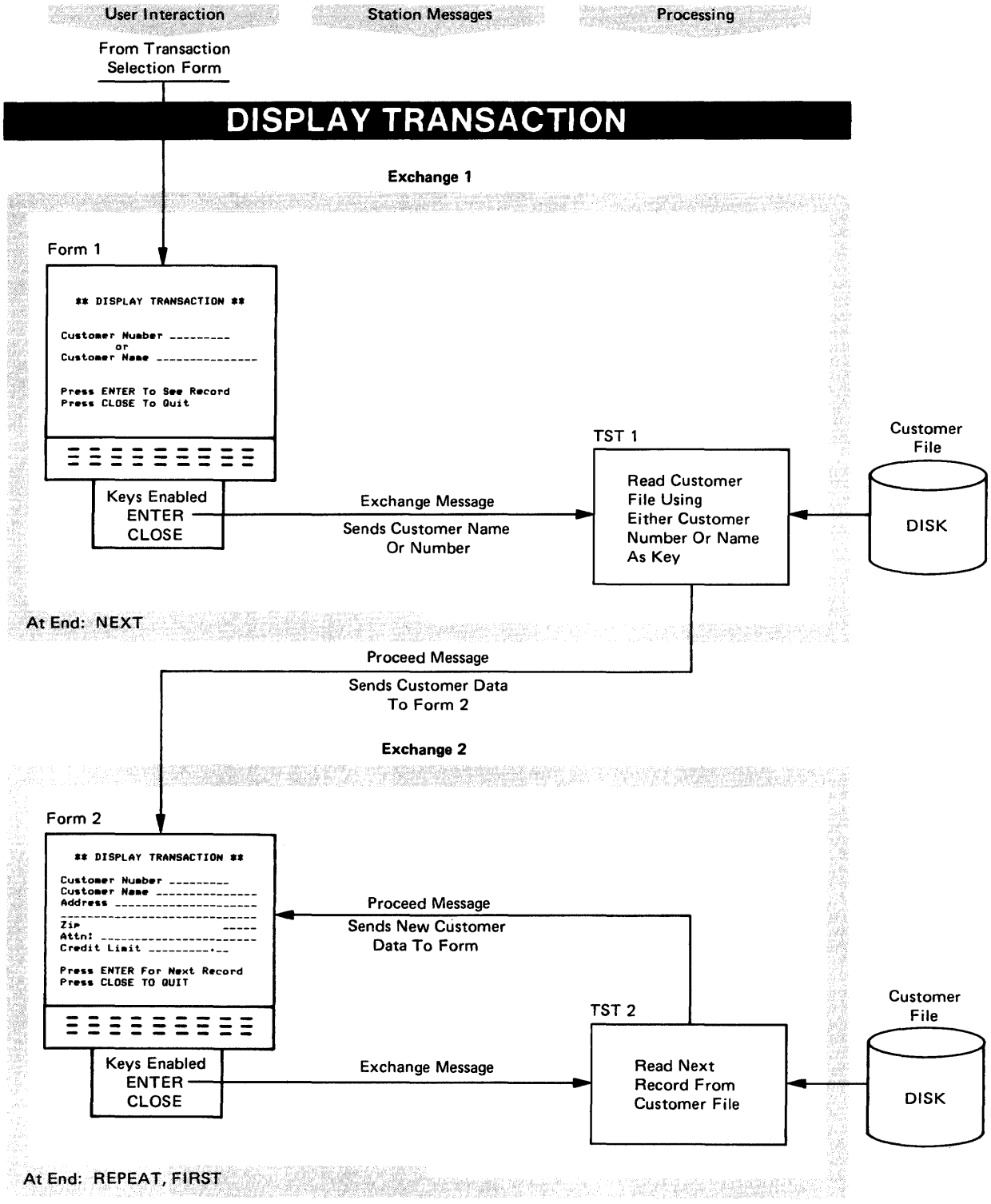


Figure 3-11 Diagram of a Display Transaction

3.2.4 Additional Definition Files

In addition to the TRADEF file, several other files are used to define a transaction processor. These are:

TPDEF (Transaction Processor Definition). This file contains general system parameters for all TPs. The TPDEF file contains records for each TP on the TRAX system.

STADEF (Station Definition). This file contains a list of all stations for a given TP such as TST stations, terminal stations, and the like. (Several other types of station types are mentioned in the next chapter.)

FILDEF (File Definition). This file lists all the data files used by the application program and provides information on file organization (sequential, relative, or indexed), record size and so on for each.

WORDEF (Work Class Definition). This file sets a series of work classes. The given work class, "CLERK" for example, is authorized to perform certain transactions. Another work class, say "SUPER," is authorized to perform another (perhaps greater) number of transactions. See the section on system security in Chapter 4.

AUTDEF (Authorization Definitions). This optional file works in conjunction with the work class file and is used to assign a user number, a password, and one or more classes to a given individual. See the section on system security in Chapter 4.

CHAPTER 4

SPECIAL TRAX CAPABILITIES

Chapter 3 developed the basic system concepts necessary to understand TRAX. This chapter builds on these concepts and presents a number of features of the TRAX system which increase its overall flexibility.

4.1 ADVANCED SYSTEM FEATURES

In the prior chapters, several special TRAX features and capabilities have been omitted, or only mentioned, while developing a general understanding of the system. These special features include:

- Record locking
- System and user workspace
- Staging
- Journalling
- Transaction logging
- Exchange Recovery
- Output-only stations
- Additional stations
- Batch processing
- Communication between transaction processors
- Links between TRAX and other computers

The following sections discuss each of these features.

4.1.1 Record Locking

To understand record locking and the advantages of this feature consider the following example. Two users on different terminals make changes to the same file and record. One changes the customer's name, and another changes the address of the same customer who had moved. Both press ENTER, assuming that their changes will be made. Both transaction instances rewrite the original record, but the changed record written first (name) is destroyed by the changed record written later (address). As a consequence the customer's new address, is lost and the integrity of the data base is corrupted.

This problem is solved in TRAX by the record locking feature. When reading a record with the intention of changing and rewriting it, the application programmer may specify a "read with lock." This means that that particular record cannot be read for purposes of updating it until the record is again unlocked. This feature applies to a single record and not to the whole file. The record locking feature and is supported by both COBOL and BASIC-PLUS-2.

4.1.2 System and User Workspace

The concept of an exchange message, specifically a message which passes data from a terminal station to a routing list of TSTs, is illustrated in Figure 3-5. Other elements associated with an exchange message were not shown at that time to simplify the diagram. These elements are shown in Figure 4-1.

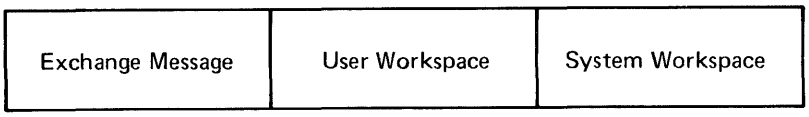


Figure 4-1 Structure of the Transaction Slot

When a user presses ENTER and the system sends an exchange message, the exchange message is a part of a larger structure called the transaction slot. The word "slot" is used here in the same sense as a slot into which you place a coin; that is, a slot is a predefined space into which something is put. In particular, the system maintains three areas in the transaction slot: the exchange message, the transaction workspace, and the system workspace.

As previously indicated the exchange message consists of data entered at a terminal and sent by the system when the user presses ENTER or another user function key. The format of the exchange message for a particular exchange is specified in the form for that exchange. It is then up to the application programmer to use the same format in a given TST that will handle the data.

The exchange message along with the entire transaction slot associated with that message is sent to the routing list of TSTs or other stations. The two other sections of the transaction slot are the transaction workspace and the system workspace. The system workspace cannot be accessed directly by the application programmer. It is used to maintain systems-related information concerning a given transaction instance.

Conversely, the transaction workspace, is available to the application programmer to use in any manner desired. It exists for the entire time that the transaction instance is active, and can be used to pass data between TSTs, or for any other purpose.

4.1.3 Staging

Staging in the TRAX system is the delay of each update to a file until the end of the transaction instance requesting the update.

If for any user or system-initiated reason the transaction instance is aborted, then the updates will not occur and the data file remains intact without cleanup operations. Staged records are stored in the system workspace to await the end of the transaction instance.

Naturally, records that are staged must be locked to prevent simultaneous update (see Record Locking). In addition, the storage of the updates utilizes system resources. TRAX allows staging to be specified on a file basis, thus limiting use of system resources to the minimum level required.

As an example of an application of staging, consider the following situation arising in an order processing department for a toy manufacturer. A store owner calls up and orders a number of different types of toys, some of which are on back-order. When the salesperson is entering the order, the store owner decides that the toys are too expensive and too many are back-ordered, so he cancels the order.

Without staging, at this point there could be records for this order in several files: items back-ordered, inventory items to be picked, and an entry in the invoice file. A tedious examination of the file would have to be made and each record created for this transaction deleted. With staging, however, the user need only press the ABORT key on the terminal, since no records have been physically updated, the data files remain intact.

4.1.4 Journalling

In many applications, it is extremely important to insure that system failures do not impact the data base. For example, a hardware disk failure might cause the contents of the disk to be destroyed. In such situations it is important to be able to reconstruct the data files up to the time of failure. In general, this situation is handled by copying files onto another peripheral device at periodic intervals. (Refer to the *TRAX System Manager's Guide* for details on this procedure.) But in such situations, the problem remains: how to reconstruct the changes made to a file since the last backup copy was made.

In TRAX, this problem may be solved by specifying the journalling feature. When journalling is specified, information necessary to reconstruct the results of each transaction are written onto another peripheral device such as a tape drive. A system-supplied utility program can then use the tape to reconstruct the files up to the point where failure occurred.

Journalling is accomplished by writing the transaction slot as it exists at the end of the transaction onto the journal device. Files whose integrity must be protected by journalling also require staging. As a consequence when journalling is specified, you also get staging. At the end of a transaction, the transaction slot contains a copy of all records in that file to be added or changed. This provides the necessary data to be able to reconstruct the files.

4.1.5 Transaction Logging

In many applications, it may be important to the application designer to provide an audit trail, or some way to reconstruct the processing performed by a given transaction processor. TRAX provides the application designer with a system program to create an audit trail by writing any specified data to the journal medium. In addition, the designer may assign a given alphabetic character to each type of log entry. This character may be used later to identify the type of entry.

To read the transaction logging file TRAX provides a utility program called "SHOLOG" which is capable of selecting and printing log records, SHOLOG is also capable of producing an RMS compatible file of selected log records.

4.1.6 Exchange Recovery

Exchange recovery provides a method of restarting an exchange without requiring subsequent input of the user data being processed by the exchange. Before the exchange begins, a copy of the exchange message is saved on disk. In the event restart is requested by a TST, then the system fetches the message and begins the exchange processing again.

Exchange recovery is particularly effective in recovery from resource interlock problems such as a record required by a transaction instance is already locked by another transaction instance. Since exchange recovery requires additional disk operations and therefore reduces system throughout, its use should be limited to "must" situations.

4.1.7 Output-only Stations

So far two types of TRAX stations have been described: TST stations and terminal stations. This section describes another type of terminal station the output-only station.

In TRAX the LA-180 output-only terminal is assigned as an output-only device to serve as a hard copy station. This terminal can be used to print reports on-line. Such a "report" consists of a form specially coded for the output-only terminal, and is typically used for relatively short data items such as a stock picking list or a hard-copy version of a record displayed on the VT62 terminal. For longer management-type reports consisting of a number of pages of data, a job would normally be submitted to the batch processor to print the report on the line printer. The TRAX line printer cannot be used as an output-only station.

Printing a report on an output-only terminal is initiated within a TST by calling the system subroutine REPORT. The output-only terminal can be used by any transaction. If more than one report is sent to the terminal at a time, the system places additional reports in a queue to wait until the previous reports are printed.

4.1.8 Additional Stations

The TRAX system provides seven types of stations:

- Terminal stations
- TST stations
- Mailbox stations
- Submit batch station
- Slave batch station
- Master link station
- Slave link station

Terminal stations and TST stations have been discussed previously. Mailbox stations are a special type of station that allows one TST to send a message to another TST. A mailbox station will retain all messages sent to it until they are picked up by the appropriate TST, consequently, a mailbox station can be used as a depository of messages. The remaining four kinds of stations are discussed in the next two sections that follow.

4.1.9 Batch Processing

Under TRAX it is possible to have a batch-processor running in the support environment. On systems with sufficient memory, a batch processor and one or more transaction processors may run simultaneously. It is also possible to have a limited degree of communication between a transaction processor (TP) and a support-environment batch job. Two special stations are provided for this function.

A submit batch station is a station in a TP that can submit a batch job, that is, request it to be placed in the batch queue, to be processed in the support environment. This is accomplished by sending a correctly formatted exchange message containing a SUBMIT request to the submit batch station which would be one of the stations on the routing list.

A slave batch station is a station in a TP that can be instructed by a program in the support environment to run a single-exchange transaction within a TP. Thus, while a submit-batch station allows a transaction to initiate processing in the support environment, a slave-batch station allows a support environment program to initiate processing within a transaction processor.

TRAX supplies the system library program that can initiate a transaction from the support environment. (See the *TRAX Application Programmer's Guide* for details.)

4.1.10 Communication Between TPs: TRAX to TRAX and TRAX to IBM

TRAX allows for communication between two or more TPs running on different CPUs. Two types of stations perform this communication: master link stations and slave link stations.

A master link station is the initiating station. If one TP is required to start a transaction in another TP, a message must first be routed to a master link station in the initiating TP. The master link station in turn initiates a data transfer to the TP at the other TRAX system.

A slave link station in the other TRAX system is the receiving station for the data. This station then begins the transaction in the receiving CPU. After the transaction is complete at the receiving TP, data may be sent back along the same set of links to the transaction at the initiating TP. One master link station may initiate transactions through a number of slave link stations.

TRAX allows for intercommunication with another TRAX computer or with an IBM computer on which CICS is run. In TRAX-to-TRAX communication, a master link station and one or more slave link stations perform the link. In IBM-to-TRAX data communications, a master link station serves as a link on the TRAX end and CICS serves as a link on the IBM end. See the *TRAX Application Programmer's Guide* for details.

4.2 TRAX SECURITY AND RELIABILITY

The security and reliability of a computer system are of prime importance. A system must be secure to prevent it from being used in unauthorized ways or by unauthorized people. A system must be reliable in performing its assigned jobs, without loss of processing ability or data stored on the system. The TRAX transaction processing system provides a number of features, described below, which make it a secure and reliable system for a wide range of data-processing applications.

4.2.1 Application Terminals and Support Terminals

In the TRAX system, the terminals assigned to transaction processing are called application terminals. Application terminals are kept completely segregated from support terminals which are used to create programs and communicate with the operating system. This restriction is strictly enforced; the system even uses separate interface devices for application terminals and support terminals. Therefore, it is, impossible for any terminal or other device connected through an application terminal interface to gain access to the TRAX operating system. As described in Chapter 2, the operation of an application terminal is strictly defined by the TP. The user is prevented from performing actions other than those specifically provided for by the application programmer.

Support terminals provide access to the operating system. However, they are used only for program development and system management; they need never be placed in a location where they would be accessible to anyone except the data processing staff. Furthermore, a production TRAX system requires only one or two support terminals that could reside in a machine room. Such placement would provide even greater security.

4.2.2 Terminal and User Authorizations

The TRAX system is designed to maintain user and terminal authority files, for authorization to run individual transactions. A given terminal may be assigned one set of transactions. A given user may be authorized to use another set of transactions. If a user tries to run a transaction he is not authorized to use, he is denied access to that transaction.

The system maintains two definition files on user authorization, the work class file and the authorization file. The work class file contains a series of work classes and a set of possible transactions for each. For example, the work class "CLERK" may be assigned one set of transactions, while the work class "MANAGR" may be assigned a different set of transactions.

The defined work classes can be used in two ways. A terminal may be assigned a certain work class. Here, a sign-on procedure might be unnecessary. In this case the terminal would be located in a place where only people qualified to perform those transactions would have access to it.

In addition, each user authorization may have one or more work classes assigned to it. This information is stored in the authorization file. This is the file that stores the user identification and password mentioned under "sign-on" below. A user may sign-on to a terminal previously assigned to a given work class and by signing on, may alter the number or type of transactions permitted to be performed.

This hierarchy of authorization has two purposes: to provide the application designer with the ability to design a system that exactly matches the variations of processing needed for the particular application, and to maintain a high degree of security at each point in that processing.

4.2.3 Initial Transaction Selections

Another way of providing system security is to define the system in such a way that, for individual terminals, the transaction selection form does not appear, only the initial form of a preassigned transaction appears. This means that for that terminal, only that transaction may be performed. This option might be useful for a situation where a person is responsible for one transaction, and there is no security problem involved in running that transaction. This arrangement would provide security by not allowing any other transactions to be run at that terminal.

4.2.4 The Sign-on Transaction

The TRAX system also provides security through a sign-on procedure. TRAX provides prewritten sign-on and sign-off transactions. In addition, for certain unusual applications, users can modify these transactions by adding their own TSTs.

A normal sign-on procedure is as follows. In a typical application, when a terminal operator turns on a terminal, the first form displayed is the Transaction Selection Form. Two of the transactions on the form will be SIGNON (sign-on) and SIGNOF (sign-off). This situation is illustrated in Figure 4-2. The Transaction Selection Form also contains a number of other possible transactions.

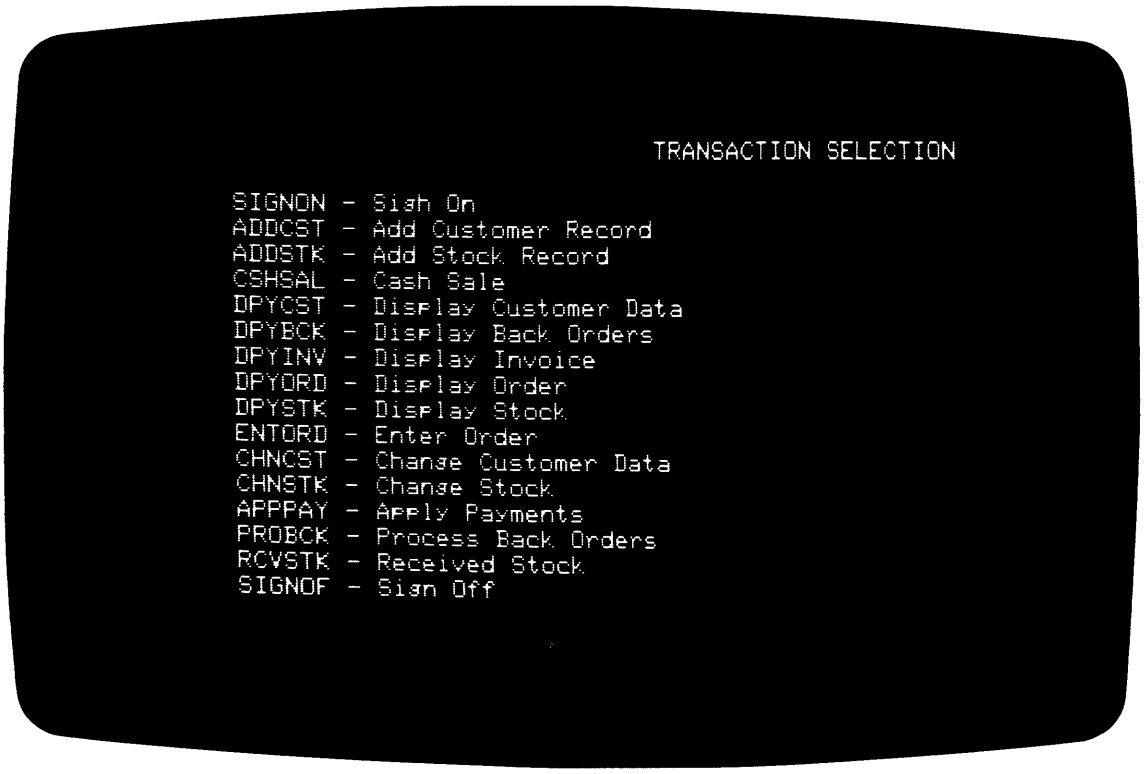


Figure 4-2 Transaction Selection Form Containing Sign-On and Sign-Off

If a user who is not signed on attempts to select any transaction other than SIGNON, the system will respond with an error message saying that he is not authorized to use that transaction. To sign on, the user selects the SIGNON transaction and presses ENTER. The sign-on form shown in Figure 4-3 will appear on the screen.

To complete the sign-on procedure, the user merely types his name, or other user identification, on the top line, and a password consisting of a prearranged group of numbers or letters on the second line. The password characters are not displayed on the screen as typed to prevent another user from learning that user's name and password. After signing on, the user can use any transaction for which he has authorization.

To sign-off, the user selects the SIGNOF transaction and presses ENTER. The sign-off form appears on the screen displaying the message "Press Key Enter to Sign Off." The user merely presses ENTER and is signed off.

4.2.5 Reliability

The TRAX system has a number of features that provide for system reliability. Many of these were described in previous sections. For example, record locking and staging are designed to ensure the integrity of the data base. Journalling and transaction logging ensure data backup in case of system

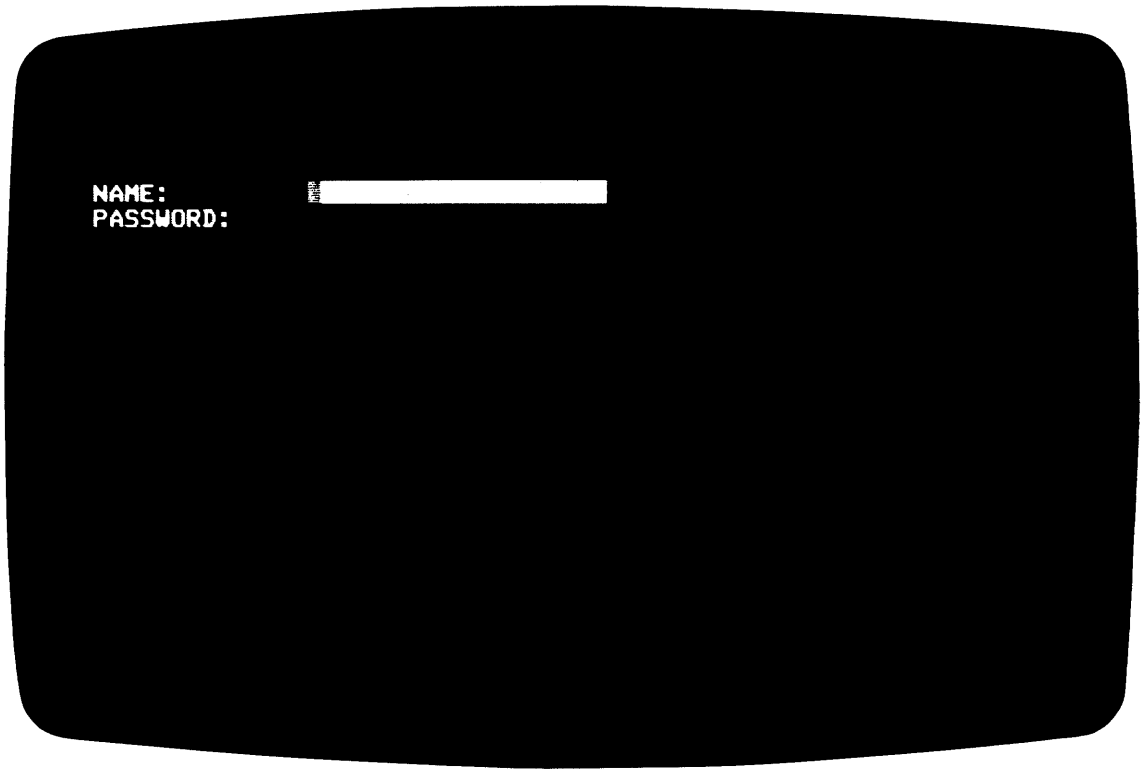


Figure 4-3 The Sign-On Form

failure. In addition, under power failure recovery, another system reliability feature, the system automatically restarts itself after a power failure or other system problems. All of these features ensure that the TRAX system maintains the highest possible degree of reliability.

4.3 TRAX DATA MANAGEMENT SYSTEM

The TRAX Data Management Service (RMS) is a system resource that can be used both by a given transaction processor and by programs in the support environment. RMS supports three types of files: sequential, relative, and indexed. Each of the three types of files is described below.

4.3.1 Sequential Files

In a sequential file records are placed one after the other in a continuous sequence. To read a given record in a sequential file, you must first read every record prior to the record being accessed. Further, to read a record before the current one, you must first close and then reopen the file.

When writing a sequential file, you can only write a record following all those previously written. In other words, you cannot insert a record into the middle of a file.

4.3.2 Relative Files

A relative file is a file divided into a sequence of numbered, fixed-size cells. The cell size is determined by the size of the largest record to be stored in the file. These cells may contain records, but there is no requirement that a given cell will contain a record. Empty cells may be interspersed with cells containing records.

Each cell from the first to the last has a number. Since cell numbers are unique, they may be used to identify a given record. In a relative file, the record may be addressed by number. On a disk file it is possible to read record 27 without having to read all the previous records. Further, it is possible to write a record in an empty cell in the middle of the file. A relative file can also be written and read in the same manner as a sequential file.

4.3.3 Indexed Files

In many data processing situations, it is desirable to store records in a file and then be able to locate a given record based on certain data contained within the record itself. For example, in an employee file it is useful to be able to identify each record in the file by the badge number or employee name.

A useful file type for this situation is the indexed file. Access to such a file is controlled by an index maintained by RMS. In a single-key indexed file, all records in the file are accessed according to the value of a certain specified field in the record known as a key.

Assume that the key is the employee name. Then RMS will construct an index based on employee names. For every record in the file, the index will contain the employee name for that record and the location of that record in the file. RMS will be able to locate the record for employee ABLE, the record for employee JONES, the record for employee SMITH, and so on.

Notice that RMS does not care what is stored in the field used as the key. Numbers, letters, special characters or any combination of these can be used. RMS merely constructs an index based on whatever is contained within the specified field.

All indexed files must have at least one key, known as the primary key. In addition, the file may have have a number of alternate keys. Assume that the employee's badge number is an alternate key. RMS would construct two indexes, a primary index and an alternate index. Any record in the file can be accessed either by the employee name for that record or the badge number for that record.

In TRAX, a requirement for the primary key is that no two records in the file can have the same key. For the alternate key, however, duplicate keys are allowed. In the example shown, it would probably make sense to interchange the two keys in order to make the badge number the primary key and the employee name the alternate key. A badge number is normally unique, but there could be two employees with the same name.

Under RMS, indexed files may be maintained only on disk. To use an indexed file, it is necessary to create an empty structure or shell before any records can be added. The RMS utility program RMSDEF is used for this process. Records can be added to the file either by another RMS utility, or by a user program which can be either a TST or a program in the support envirionment. Notice that the add transaction shown in Chapter 2 contains an example of such a TST.

An application program may use indexed files in a variety of ways. For a given key, either employee name or badge number, a program can read, write, change or delete a record; for example, the record for employee "Jones" or for badge number "11733").

In addition, an indexed file can be used in a sequenced manner because the indices are arranged in alphabetic or numeric order. After a read based on badge number 11781, a program may perform a statement of the form "read next record." If this is done, this record with badge number 11782 or the next sequential record in the file is returned. Successive "read next record" statements read records with sequentially ordered badge numbers.

Similarly if the "read next record" statement follows a read based on employee name JONES, the statement would return the next record in alphabetic order by customer name – perhaps JOYCE. Successive "read next record" statements would then read records in alphabetic order by customer name.

CHAPTER 5

THE TRAX SUPPORT ENVIRONMENT

The TRAX support environment is the non-transaction processing part of the TRAX system where program development and system maintenance functions are performed. In a system which supports simultaneous transaction processing and full support environment services, the support environment user may be unaware that the system is simultaneously processing transactions, except for slower response times. In systems without a full support environment, only required transaction processing utilities may be executed. To the support environment user, the system seems to be a normal multiprogramming system for creating and updating files, developing and running programs, and the like.

Application terminals which are used for processing transactions cannot be used for program development in the support environment. To create a transaction processor, the application programmer first develops all the components (Forms, TSTs, etc.) on a support terminal and then starts the transaction processor. The TP then runs on the application terminals.

The purpose of this chapter is to provide a brief introduction to the services available to support environment programmers or other users. For more information, see the *TRAX Support Environment User's Guide.*

5.1 THE KERNEL OPERATING SYSTEM

The Kernel operating system is a multitasking operating system that can control and run many tasks simultaneously. Tasks are created by compiling and linking a program. Tasks can be stored on disk until they are needed, and then quickly loaded into memory and executed.

5.2 THE TRAX FILE SYSTEM

TRAX provides the user with a number of file-handling capabilities. For example, a user can create a file at a terminal, and an application program can read from and write onto a file. Such files are managed through TRAX Data Management Services (RMS). RMS handles sequential, relative and indexed files. In the support environment, RMS routines are linked directly with the pertinent task and therefore are non-shareable. Any user program which accesses a file does so through RMS. Further, any command entered at a support terminal which accesses a file does so through RMS.

5.3 THE DIGITAL COMMAND LANGUAGE

The DIGITAL Command Language (DCL) is a convenient means of communicating with the operating system. Using common English words such as RUN, COPY, EDIT, DIRECTORY and DELETE, you can request a variety of services including editing, file handling, compiling, linking, running, printing and many others.

DCL supports two levels of users: privileged and nonprivileged. Privileged users can execute all DCL commands, some of which can affect the way the system performs for other users. Nonprivileged users can execute a subset of these commands that provide all normal processing capability. The commands available to privileged users are defined in the *TRAX System Manager's Guide.* The commands for nonprivileged users are defined in the *TRAX Support Environment User's Guide.*

5.3.1 Spooling Files to the Line Printer

The line-printer spooler allows you to print copies of files on the line printer without having to wait at the terminal until the line printer is free. To print a file, type PRINT and the file specification, and the file is immediately placed in a queue to be printed. The TRAX system provides several commands to show a file's location in the queue, and to modify the queue if desirable.

5.3.2 Other DCL Commands

There are other DCL commands which perform the functions listed below:

ABORT – stop a running task or a command.
ALLOCATE – assign a device, such as a disk file, to a user for that person's exclusive use.
BASIC – compile a BASIC-PLUS-2 source program.
COBOL – compile a COBOL source program.
DELETE – delete a file.
DIRECTORY – list a selected group of file names and sizes.
EDIT – call the system editor to edit a file.
HELP – provide information on one of the commands.
LOGIN – provide access to the system.
MESSAGE – send messages to other users on the system.
RENAME – assign a different name to a file.
SET – set a number of system parameters.
SHOW – show the value of these same parameters.

The complete set of DCL commands are described in the *TRAX Support Environment User's Guide.*

5.4 PROGRAMMING LANGUAGES

The TRAX system supports COBOL and BASIC-PLUS-2.

5.4.1 COBOL

The COBOL that runs on the TRAX system is DIGITAL's version 3.5 COBOL defined as a large subset of ANSI 1974 COBOL. There are also some extensions for use only in TSTs. Some of these are the "READ WITH LOCK", "WRITE WITH LOCK", "REWRITE WITH LOCK", and "UNLOCK" statements. These statements help solve problems involving simultaneous updates to a file.

The TRAX/COBOL product includes:

- COBOL compiler and run-time system
- RFRMT source program reformat utility
- CBLMRG COBOL ODL merge utility
- Language extension for record locking

5.4.2 BASIC-PLUS-2

Description:
BASIC is a conversational programming language which uses simple English-like statements to describe a procedure. The BASIC-PLUS-2 language is a superset of the RSTS/E BASIC-PLUS and Dartmouth BASIC languages. The BASIC-PLUS-2 compiler runs under the TRAX operating system, and produces object modules which can be linked by the operating system's linker.

BASIC-PLUS-2 language features include:

- CALL statement – allows interface to separately compiled BASIC-PLUS-2 subroutines. The subroutines can be called by name and passed several arguments.
- COM or COMMON statement – allows data to be passed between program segments.
- RECORD I/O using the operating system's RMS file structure – allows logical record manipulation and record locking.
- Multi-key Indexed Sequential Access Method using the RMS indexed file organization access methods.
- Interactive debugging commands to speed program development.

5.5 TRAX SYSTEM UTILITIES

The TRAX system provides a variety of utilities. Among these are SORT, DATATRIEVE, and various transaction processing utilities.

5.5.1 SORT

The SORT utility allows you to read any input file, sort the contents, and write sorted data on to an output file. The sort may be based on any set of keys for a given record. If you do not wish to sort your data base, you can use sort to extract selected portions of a record from the data base, and then sort those records according to predetermined keys. SORT handles any file organization supported by RMS: sequential, relative or indexed. Since an indexed file is already sorted through the index, SORT can be used to sort it by keys different from those maintained by the index. SORT can only be used in the support environment.

5.5.2 DATATRIEVE

DATATRIEVE is an interactive data-interrogation and report-writing utility for use in the support environment. Although the same functions could be performed in higher-level languages, DATATRIEVE command sequences can be written on line and tested interactively. No compiling and linking are required. The commands are simple to use, and are forgiving of errors.

DATATRIEVE can be used to interrogate a data base, select certain information, sort the information and print it in a clear, usable form. In addition, commonly used sequences of code may be retained in the files and used by other DATATRIEVE users later.

5.5.3 Transaction Processing Utilities

There are several utilities to support the definition and maintenance of transaction processors. For example, the definition utilities TPDEF, TRADEF, STADEF, FILDEF, WORDEF and AUTDEF described in Chapter 3 are in this category.

In addition, TRAX provides the following utility programs for transaction processors:

ARCHIVE – allows you to back up files that is, make copies in case of device failure and to restore the original files in the event of a failure.
RECOVER – allows you to make all changes to a backup file since the last time it was backed up, provided the transactions were journalled.
SHOLOG – allows you to find and print certain records in a file created by a user-logging procedure.
IPCIRL – allows you to control execution of a transaction processor. Commands include INSTALL, START, STOP and REMOVE.
IPMOD – allows you to change certain parameters of a running transaction processor, for example, to connect or disconnect a terminal station.
SERLOG – logs all software error messages such as messages to indicate a system-aborted transaction. The messages are written to a file and optionally to a support terminal.
IPSTAT – collects samples of key system statistics at predetermined intervals and records these on a file. These statistics can then be used to determine system loading and to make alterations to the system to improve performance.
STAREP – analyzes and displays in a meaningful format the statistical data collected by TPSTAT.

In addition, there are a number of other utilities described in the *TRAX Application Programmer's Guide* and *TRAX System Manager's Guide.*

5.6 BATCH PROCESSING IN THE SUPPORT ENVIRONMENT

The TRAX system provides a full and flexible batch-processing capability. The user submits a job for batch processing at any support terminal by typing the SUBMIT command. This causes the system to place the job in the batch queue to be executed in its turn. If desired, the SUBMIT AFTER form of the command can be used to initiate a job that is to be run after a certain time.

Another option allows a PRIORITY to be specified, causing the current job to be run sooner or later than other batch jobs in the queue. In addition, there are a number of commands that allow the programmer or terminal user to display the current batch queue and make changes to it.

APPENDIX A
TRAX USERS AND THE MANUAL SET

POTENTIAL USERS OF THE TRAX MANUAL SET

In any given TRAX installation, there are a number of ways individuals can use the system. These are summarized in the table below. The TRAX manual set is also described in this appendix.

TRAX System Users

User	User Function and Information Needs
Terminal User	Person who runs transactions at an application terminal. Needs to know clerical procedures and how to operate the application terminal.
Data-Entry Manager	Person who supervises terminal users. May need a complete understanding of the system or only the minimum required to instruct employees in how to perform their jobs.
System Manager	Person who is responsible for the system in general and for individual user accounts on the system in particular. The system manager mounts tapes, performs routine maintenance, makes backup copies of important files, and generally oversees computer operation. Needs a general understanding of the system, plus specific procedures to perform individual tasks.
Application Designer	Person who designs the transaction processing application, and delegates programming of that application to the application programmer. Needs to understand the system operation in detail and how to implement a given application on the TRAX system. Also needs to understand various TRAX features such as staging and exchange recovery, and how design tradeoffs affect system performance.
Application Programmer	Persons who code a given transaction processing application based on specifications and directions provided by the application designer. Needs to know general information about the support environment, plus specific information about coding forms, coding transaction step tasks, and running TRAX utility programs.

THE TRAX MANUAL SET

The TRAX manual set consists of 19 documents. Each is described, along with its intended audience, in the following table.

The TRAX Manual Set

Manual and Order No.	Manual Content and Audience
	SYSTEM MANUALS
Introduction to TRAX (Order No. AA-D327A-TC)	Provides a general understanding of the TRAX system. Intended audience: all users of the TRAX transaction processing system, including those who wish only a brief introduction to the system. (Chapters 1 and 2)
TRAX Application Designer's Guide (Order No. A-D328A-TC) (Manual consists of two parts, Part I and Part II)	Provides a detailed and thorough understanding of the structure and operation of a TRAX transaction processing application. In addition, provides a method for developing a given transaction processing application. Intended audience: application designer.
TRAX Application Programmer's Guide (Order No. AA-D329A-TC)	Provides the specific information needed by an application programmer to code, debug, and run a transaction processing application. Intended audience: application programmer and application designer.
TRAX Application Terminal Language (ATL) Reference Manual (Order No. AA-D330A-TC)	Describes how to code forms using the Application Terminal Language (ATL). Intended audience: application programmer and application designer.
TRAX Support Environment User's Guide (Order No. AA-D331A-TC)	Describes how to perform normal support environment functions: logging in on a terminal; creating and editing files, programs; and so on. Intended audience: all TRAX programmers.
TRAX System Manager's Guide (Order No. AA-D332A-TC)	Describes how to perform system operator functions such as startup, shutdown, and batch processing control; transaction processor management functions such as installing and starting a transaction processor; support environment management functions such as hardware error logging and running utilities. Also describes how to use the DIGITAL Command Language (DCL) for both privileged and non-privileged commands. Intended audience: system manager.
TRAX System Generation Manual (Order No. AA-D335A-TC)	Describes, step-by-step, how to run the program that creates a TRAX system which has been customized to meet the exact requirements of the particular installation. Intended audience: system manager.
	LANGUAGE MANUALS
TRAX BASIC-PLUS-2 Language Reference Manual (Order No. AA-D336A-TC)	Describes the general structure and statement format of BASIC-PLUS-TWO. Intended audience: PASIC-PLUS-2 programmer.
TRAX BASIC-PLUS-2 User's Guide (Order No. AA-D337A-TC)	Describes the specific features of BASIC-PLUS-2 as implemented on the TRAX system. Intended audience: BASIC-PLUS-2 programmer.

The TRAX Manual Set

Manual and Order No.	Manual Content and Audience
	LANGUAGE MANUALS
TRAX COBOL Language Reference Manual (Order No. AA-D338A-TC)	Describes the general structure and statement format of COBOL as implemented on DIGITAL computers. Intended audience: COBOL programmer.
TRAX COBOL User's Guide (Order No. AA-D339A-TC)	Describes the specific features of COBOL as implemented on the TRAX system. Intended audience: COBOL programmer.
TRAX MACRO Language Reference Manual (Order No. AA-D340A-TC)	Describes how to use the MACRO assembly language. Intended audience: assembly language programmer.
	UTILITY-PROGRAM MANUALS
DEC EDITOR Reference Manual (Order No. AA D347A-TC)	Describes how to use DEC EDITOR, the utility program that allows you to perform a wide variety of edits (i.e., changes) on files. Intended audience: All TRAX programmers and those who use the system to store and maintain documents.
TRAX Linker Reference Manual (Order No. AA-D344A-TC)	Describes how to use the system linking program. Intended audience: all TRAX programmers.
TRAX RMS MACRO Programmer's Guide (Order No. AA-D344A-TC)	Provides detailed information about RMS (Records Management Services) and shows the MACRO programmer how to access this structure directly. RMS is a record and data management system that supports several types of file organizations, including indexed files. Intended audience: MACRO programmer.
TRAX ODT Reference Manual (Order No. AA-D343A-TC)	Describes how to use the ODT (On-line Debugging Technique) utility for debugging MACRO programs. ODT provides capabilities such as running the user's program up to a selected breakpoint, making modifications and then running it again. Intended audience: MACRO programmer.
TRAX User Mode Diagnostics Reference Manual (Order No. AA-D344A-TC)	Describes how to run User Mode Diagnostics, a set of utilities to test out any given peripheral device in case of suspected errors, or as part of a preventive maintenance program. Intended audience: system manager.
TRAX SORT Reference Manual (Order No. AA-346A-TC)	Describes how to use the SORT utility to sort a file according to specified keys. A number of different types of sorts are available. Intended audience: any programmer, operator or other system user who needs to sort files.

The TRAX Manual Set

Manual and Order No.	Manual Content and Audience
	UTILITY-PROGRAM MANUALS
TRAX DATATRIEVE User's Guide (Order No. AA-D347A-TC)	Describes how to use DATATRIEVE, an interactive utility used to access (and optionally modify) a data base, and print reports based on that data base. The commands allow for simple generation of headings, selecting given records and items in a record, sorting data, and printing final reports. Intended audience: any data base user.

Figure B-1 Form One of the Display Transaction

```
DEFAULT
        CLEAR = " "
        ENABLE = CLOSE

FORM
        SPLIT=8                                 18 lines of display area

DISPLAY = 3,12
        VALUE = "Customer Master File Subsystem - Display Customer Transaction"
        LENGTH = 60
        ATTRIBUTE = REVERSE,NOBLANK

DISPLAY = 5,1
        LABEL = REPLY.TEXT.A
        LENGTH = 80

DISPLAY = 6,1
        LABEL = REPLY.TEXT.B
        LENGTH = 80

PROMPT = 1,1
        VALUE = "Customer Number"

INPUT = .,20
        LABEL = CUST.NO
        LENGTH = 6
        CLEAR = "0"
        ATTRIBUTE = REVERSE,RIGHT

PROMPT = .+2,10
        VALUE = "-- OR --"

PROMPT = .+2,1
        VALUE = "Customer Name"

INPUT = .,20
        LABEL = CUST.NAME
        LENGTH = 30
        ATTRIBUTE = REVERSE

PROMPT = 15,1
        LABEL = KEY.PROMPT
        LENGTH = 80
        VALUE = "Function Keys:  ",
                "ENTER to see customer record, ",
                "CLOSE to quit function"
        ATTRIBUTE = REVERSE

MESSAGE = 1
        VALUE =
                CUST.NO,
        CUST.NAME

REPLY = 2
        WRITE = REPLY.TEXT.A, REQUEST(1,80)
        WRITE = REPLY.TEXT.B, REQUEST(81,80)

END
```

Figure B-2 ATL Source Code for Form One

```
00001        IDENTIFICATION DIVISION.
00002
00003        PROGRAM-ID. TSTEP.
00004        AUTHOR.  DAVID-BRITT.
00005        INSTALLATION. DEC.
00006        DATE-WRITTEN. 12-DEC-77.
00007        DATE-COMPILED. TODAY.
00008
00009            22-MAR-78 .
00010        ENVIRONMENT DIVISION.
00011
00012        CONFIGURATION SECTION.
00013            SOURCE-COMPUTER.  PDP-11.
00014            OBJECT-COMPUTER.  PDP-11.
00015
00016        INPUT-OUTPUT SECTION.
00017        FILE-CONTROL.
00018            SELECT CUSTOM ASSIGN TO "CUSTOM.DAT"
00019                    ORGANIZATION IS INDEXED
00020                    ACCESS MODE IS DYNAMIC
00021                    RECORD KEY IS CUSTOMER-NUMBER
00022                    ALTERNATE RECORD KEY IS CUSTOMER-NAME
00023                     WITH DUPLICATES
00024                    FILE STATUS IS CUSTOMER-FILE-STATUS.
00025
00026        DATA DIVISION.
00027
00028        FILE SECTION.
00029        FD  CUSTOM
00030            LABEL RECORDS ARE STANDARD
00031            VALUE OF ID IS CUSTOMER-CHANNEL-NUMBER
00032            DATA RECORD IS CUSTOMER-FILE-RECORD.
00033
00034        01  CUSTOMER-FILE-RECORD.
00035            03  CUSTOMER-NUMBER                    PIC X(6).
00036            03  CUSTOMER-NAME                      PIC X(30).
00037            03  ADDRESS-LINE-1                     PIC X(30).
00038            03  ADDRESS-LINE-2                     PIC X(30).
00039            03  ADDRESS-LINE-3                     PIC X(30).
00040            03  ADDRESS-ZIP-CODE                   PIC 9(5).
00041            03  TELEPHONE-NUMBER                   PIC 9(10).
00042            03  ATTENTION-LINE                     PIC X(20).
00043            03  CREDIT-LIMIT-AMOUNT                PIC 9(10)V99.
00044            03  CURRENT-BALANCE                    PIC 9(10)V99.
00045            03  PURCHASES-YTD                      PIC 9(10)V99.
00046            03  NEXT-ORDER-SEQUENCE-NUM            PIC 9(4).
00047            03  NEXT-PAYMENT-SEQUENCE-NUM          PIC 9(4).
00048
00049        WORKING-STORAGE SECTION.
00050        01  CUSTOMER-CHANNEL-NUMBER          PIC X(11)
00051                VALUE IS "CUSTOM/CH:3".
00052        01  CUSTOMER-FILE-STATUS             PIC XX.
00053        01  FILE-STATUS-WORD                       PIC XX.
```

Figure B-3 TST FIND: COBOL Identification, Environment and Data Divisions (1 of 2)

```
00054        01  BUFFER-SIZE                          PIC 9999 COMP.
00055        01  STATUS-WORDS.
00056            03  STATUS-WORD-1                    PIC S9(4) COMP.
00057            03  STATUS-WORD-2                    PIC S9(4) COMP.
00058        01  REPLY-NUMBER                         PIC 9999 COMP.
00059        01  REPLY-MESSAGE-BUFFER.
00060            03  REPLY-MESSAGE-TEXT               PIC X(80).
00061            03  REPLY-FILLER                             PIC X(16)
00062            VALUE IS "FILE STATUS WORD".
00063            03  RMB-FSW                                  PIC X(4).
00064            03  RMB-FILE-NAME                            PIC X(60).
00065        01  EDIT-FIELD                           PIC Z,ZZZ,ZZZ.99.
00066
00067        LINKAGE SECTION.
00068        01  EXCHANGE-MESSAGE.
00069            02      EM-INPUT-FORM-DPCUS1.
00070                    03  EM-CUSTOMER-NUMBER               PIC X(6).
00071                    03  EM-CUSTOMER-NAME                 PIC X(30).
00072
00073        01  TST-WORKSPACE.
00074            02      WS-CUSTOMER-FILE-RECORD.
00075                    03  WS-CUSTOMER-NUMBER               PIC X(6).
00076                    03  WS-CUSTOMER-NAME                 PIC X(30).
00077                    03  WS-ADDRESS-LINE-1                PIC X(30).
00078                    03  WS-ADDRESS-LINE-2                PIC X(30).
00079                    03  WS-ADDRESS-LINE-3                PIC X(30).
00080                    03  WS-ADDRESS-ZIP-CODE              PIC 9(5).
00081                    03  WS-TELEPHONE-NUMBER              PIC 9(10).
00082                    03  WS-ATTENTION-LINE                PIC X(20).
00083                    03  WS-CREDIT-LIMIT-AMOUNT           PIC X(12).
00084                    03  WS-CURRENT-BALANCE               PIC X(12).
00085                    03  WS-PURCHASES-YTD                 PIC X(12).
00086                    03  WS-NEXT-ORDER-SEQUENCE-NUM       PIC 9(4).
00087                    03  WS-NEXT-PAYMENT-SEQUENCE-NUM     PIC 9(4).
00088
```

Figure B-3 TST FIND: COBOL Identification, Environment and Data Divisions (2 of 2)

```
00089          PROCEDURE DIVISION USING EXCHANGE-MESSAGE, TST-WORKSPACE.
00090
00091          DECLARATIVES.
```

Note: The DECLARATIVES Section Is Shown In Figure B-5

```
00200          END DECLARATIVES.
00201
00202          TST-PROCESSING-SECTION SECTION.
00203          TEST-FIELDS.
00204              OPEN INPUT CUSTOM.
00205              IF CUSTOMER-FILE-STATUS IS GREATER THAN "09"
00206              GO TO END-PROGRAM.
00207
00208              IF EM-CUSTOMER-NUMBER IS GREATER THAN "000000"
00209              GO TO READ-BY-NUMBER.
00210
00211              IF EM-CUSTOMER-NAME IS GREATER THAN SPACES
00212              GO TO READ-BY-NAME.
00213
00214              GO TO END-PROGRAM.
00215
00216          READ-BY-NUMBER.
00217              MOVE EM-CUSTOMER-NUMBER TO CUSTOMER-NUMBER.
00218              READ CUSTOM
00219              KEY IS CUSTOMER-NUMBER.
00220              IF CUSTOMER-FILE-STATUS IS GREATER THAN "09"
00221              GO TO END-PROGRAM.
00222
00223              GO TO DISPLAY-CUSTOMER-RECORD.
00224
00225          READ-BY-NAME.
00226              MOVE EM-CUSTOMER-NAME TO CUSTOMER-NAME.
00227              READ CUSTOM
00228              KEY IS CUSTOMER-NAME.
00229              IF CUSTOMER-FILE-STATUS IS GREATER THAN "09"
00230              GO TO END-PROGRAM.
00231
00232              GO TO DISPLAY-CUSTOMER-RECORD.
00233
00234          DISPLAY-CUSTOMER-RECORD.
00235              MOVE 205 TO BUFFER-SIZE.
00236              MOVE CUSTOMER-FILE-RECORD TO WS-CUSTOMER-FILE-RECORD.
00237              MOVE CREDIT-LIMIT-AMOUNT TO EDIT-FIELD.
00238              MOVE EDIT-FIELD TO WS-CREDIT-LIMIT-AMOUNT.
00239              MOVE CURRENT-BALANCE TO EDIT-FIELD.
00240              MOVE EDIT-FIELD TO WS-CURRENT-BALANCE.
00241              MOVE PURCHASES-YTD TO EDIT-FIELD.
00242              MOVE EDIT-FIELD TO WS-PURCHASES-YTD.
00243
00244              CALL "PRCEED" USING WS-CUSTOMER-FILE-RECORD,
00245                                  BUFFER-SIZE,
00246                                   STATUS-WORDS.
00247
00248          END-PROGRAM-SECTION SECTION.
00249          END-PROGRAM.
00250              EXIT PROGRAM.
```

Figure B-4 TST FIND: Procedure Division (Minus Declaratives Section)

```
00089       PROCEDURE DIVISION USING EXCHANGE-MESSAGE, TST-WORKSPACE.
00090
00091       DECLARATIVES.
00092       I-O-ERROR SECTION.
00093           USE AFTER STANDARD ERROR PROCEDURE ON CUSTOM.
00094       CHECK-FILE-STATUS-CODE.
00095
00096           IF CUSTOMER-FILE-STATUS IS GREATER THAN "01"
00097           MOVE CUSTOMER-FILE-STATUS TO FILE-STATUS-WORD
00098           MOVE "LOGICAL FILE NAME: CUSTOM -CH3" TO
00099           RMB-FILE-NAME.
00100
00101           IF FILE-STATUS-WORD IS EQUAL TO "10"
00102                   MOVE "Reached End-of-File"
00103                   TO REPLY-MESSAGE-BUFFER
00104                   GO TO SEND-REPLY-MESSAGE.
00105
00106           IF FILE-STATUS-WORD IS EQUAL TO "21"
00107                   MOVE "Primary Key Sequence Error on WRITE"
00108                   TO REPLY-MESSAGE-BUFFER
00109                   GO TO SEND-ABORT-MESSAGE.
00110
00111           IF FILE-STATUS-WORD IS EQUAL TO "22"
00112                   MOVE "Duplicate Key Error"
00113                   TO REPLY-MESSAGE-BUFFER
00114                   GO TO SEND-ABORT-MESSAGE.
00115
00116           IF FILE-STATUS-WORD IS EQUAL TO "23"
00117                   MOVE "No Record Exists under that Key"
00118                   TO REPLY-MESSAGE-BUFFER
00119                   GO TO SEND-REPLY-MESSAGE.
00120
00121           IF FILE-STATUS-WORD IS EQUAL TO "24"
00122                   MOVE "Boundary Error on Write Statement"
00123                   TO REPLY-MESSAGE-BUFFER
00124                   GO TO SEND-ABORT-MESSAGE.
00125
00126           IF FILE-STATUS-WORD IS EQUAL TO "30"
00127                   MOVE "Unspecified I/O Error"
00128                   TO REPLY-MESSAGE-BUFFER
00129                   GO TO SEND-ABORT-MESSAGE.
00130
00131           IF FILE-STATUS-WORD IS EQUAL TO "34"
00132                   MOVE "Permanent Boundary Error on WRITE Statement"
00133                   TO REPLY-MESSAGE-BUFFER
00134                   GO TO SEND-ABORT-MESSAGE.
00135
00136           IF FILE-STATUS-WORD IS EQUAL TO "91"
00137                   MOVE "File locked by another task"
00138                   TO REPLY-MESSAGE-BUFFER
00139                   GO TO SEND-REPLY-MESSAGE.
00140
00141           IF FILE-STATUS-WORD IS EQUAL TO "92"
00142                   MOVE "Record locked by another task"
00143                   TO REPLY-MESSAGE-BUFFER
00144                   GO TO SEND-REPLY-MESSAGE.
00145
00146           IF FILE-STATUS-WORD IS EQUAL TO "93"
00147                   MOVE "REWRITE or DELETE attempted without prior
00148       -           "READ being performed."
00149                   TO REPLY-MESSAGE-BUFFER
00150                   GO TO SEND-REPLY-MESSAGE.
00151
```

Figure B-5 TST FIND: COBOL Declaratives Section (1 of 2)

```
00152            IF FILE-STATUS-WORD IS EQUAL TO "94"
00153                    MOVE "Improper operation attempted"
00154                    TO REPLY-MESSAGE-BUFFER
00155                    GO TO SEND-ABORT-MESSAGE.
00156
00157            IF FILE-STATUS-WORD IS EQUAL TO "95"
00158                    MOVE "Allocation Failure - No space on device"
00159                    TO REPLY-MESSAGE-BUFFER
00160                    GO TO SEND-ABORT-MESSAGE.
00161
00162            IF FILE-STATUS-WORD IS EQUAL TO "96"
00163                    MOVE "No buffer space - SAME AREA already in use"
00164                    TO REPLY-MESSAGE-BUFFER
00165                    GO TO SEND-ABORT-MESSAGE.
00166
00167            IF FILE-STATUS-WORD IS EQUAL TO "97"
00168                    MOVE "Unable to find file named:"
00169                    TO REPLY-MESSAGE-BUFFER
00170                    GO TO SEND-ABORT-MESSAGE.
00171
00172            IF FILE-STATUS-WORD IS EQUAL TO "98"
00173                    MOVE "Error while attempting to CLOSE file."
00174                    TO REPLY-MESSAGE-BUFFER
00175                    GO TO SEND-ABORT-MESSAGE.
00176
00177            MOVE "UNKNOWN I-O ERROR" TO REPLY-MESSAGE-TEXT
00178            MOVE FILE-STATUS-WORD TO RMB-FSW
00179            GO TO SEND-ABORT-MESSAGE.
00180
00181        SEND-REPLY-MESSAGE.
00182            MOVE 160 TO BUFFER-SIZE
00183            MOVE 2 TO REPLY-NUMBER
00184            CALL "REPLY" USING
00185                            REPLY-MESSAGE-BUFFER,
00186                            BUFFER-SIZE,
00187                            REPLY-NUMBER,
00188                            STATUS-WORDS.
00189            GO TO END-ERROR-SECTION.
00190
00191        SEND-ABORT-MESSAGE.
00192            MOVE 160 TO BUFFER-SIZE
00193            MOVE 2 TO REPLY-NUMBER
00194            CALL "ABORT" USING
00195                            REPLY-MESSAGE-BUFFER
00196                            BUFFER-SIZE
00197                            REPLY-NUMBER
00198                            STATUS-WORDS.
00199        END-ERROR-SECTION.
00200        END DECLARATIVES.
```

Figure B-5 TST FIND: COBOL Declaratives Section (2 of 2)

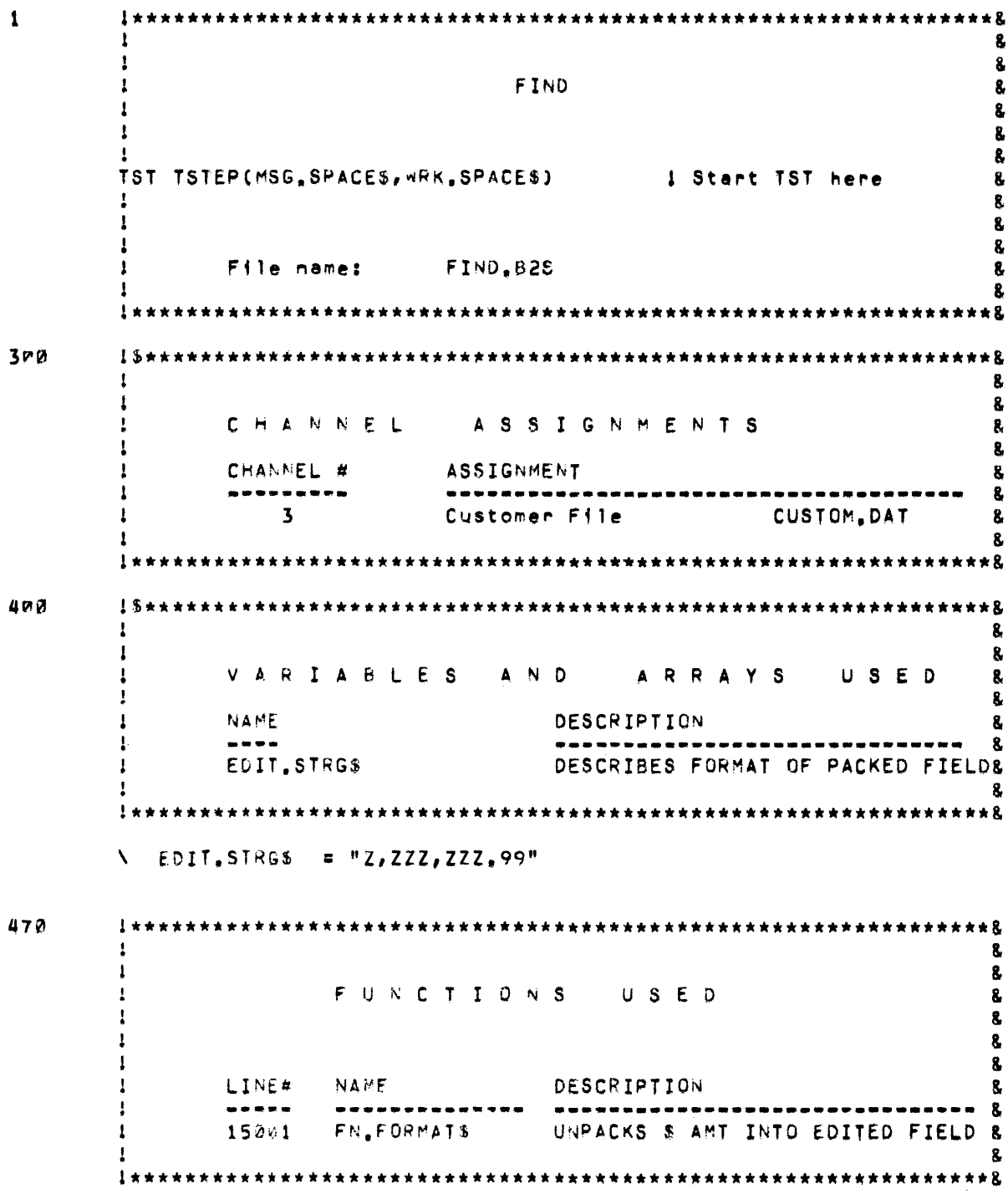


Figure B-6 The BASIC PLUS-2 TST FIND (1 of 8)

```
600     !$*****************************************************************&
        !                                                                 &
        !                                                                 &
        !       E X C H A N G E    M E S S A G E   D E F I N I T I O N    &
        !                                                                 &
        !                                                                 &
        !*****************************************************************&

620     !*****************************************************************&
        !                                                                 &
        !                                                                 &
        !               Exchange Message for Form DPCUS1                  &
        !                                                                 &
        !                                                                 &
        !                                                                 &
        \ MSGMAP        !DPCUS1!                ! Level = 02    !       &
        !                                                                 &
              EXCHANGE.MESSAGE.DPCUS1.FORM$   =  36                       &
        !                                                                 &
        \ MSGMAP        !DPCUS1!                ! Level = 03    !       &
        !                                                                 &
              EM.CUSTOMER.NUMBER$ =   6                                   &
        ,     EM.CUSTOMER.NAME$   =  30                                   &

700     !$*****************************************************************&
        !                                                                 &
        !                                                                 &
        !       W O R K S P A C E    D E F I N I T I O N                  &
        !                                                                 &
        !                                                                 &

775     !*****************************************************************&
        !                                                                 &
        !                                                                 &
        !               Working Storage Customer File Record              &
        !                                                                 &
        !                       for TST FIND                              &
        !                                                                 &
        \ WRKMAP        !CUSTOM!                ! Level = 02    !       &
        !                                                                 &
              WS.CUSTOMER.FILE.RECORD$        =  205                      &
        !                                                                 &
        \ WRKMAP        !CUSTOM!                ! Level = 03    !       &
        !                                                                 &
              WS.CUSTOMER.NUMBER$     =   6                               &
        ,     WS.CUSTOMER.NAME$       =  30                               &
        ,     WS.ADDRESS.1$           =  30                               &
        ,     WS.ADDRESS.2$           =  30                               &
        ,     WS.ADDRESS.3$           =  30                               &
        ,     WS.ZIP.CODE$            =   5                               &
        ,     WS.TELEPHONE.NUMBER$    =  10                               &
        ,     WS.ATTENTION.LINE$      =  20                               &
        ,     WS.CREDIT.LIMIT$        =  12                               &
        ,     WS.CURRENT.BALANCE$     =  12                               &
        ,     WS.PURCHASES.YTD$       =  12                               &
        ,     WS.NEXT.ORDER.NUMBER$   =   4                               &
        ,     WS.NEXT.PAYMENT.NUMBER$ =   4                               &
        !*****************************************************************&
```

Figure B-6 The BASIC PLUS-2 TST FIND (2 of 8)

```
900     !$*****************************************************************&
        !                                                                 &
        !                                                                 &
        !       D I M E N S I O N     D E C L A R A T I O N S             &
        !                                                                 &
        !                                                                 &
        !                                                                 &
        !       Lines 901-929 denote local dimension declarations.        &
        !       Lines 930-949 denote library dimension declarations.      &
        !       Lines 950-979 denote MAP statements.                      &
        !       Lines 980-999 denote IMAGE statements.                    &
        !                                                                 &
        !*****************************************************************&

950     !$*****************************************************************&
        !                                                                 &
        !                                                                 &
        !       F I L E   R E C O R D   D E F I N I T I O N S             &
        !                                                                 &
        !                                                                 &
        !*****************************************************************&


952     !*****************************************************************&
        !                                                                 &
        !                                                                 &
        !                 Customer File Record                            &
        !                                                                 &
        \ MAP   (CUSTOM)                        ! Level = 01    !         &
        !                                                                 &
                CUSTOMER.FILE.RECORD$   = 205                             &
        !                                                                 &
        \ MAP   (CUSTOM)                        ! Level = 03    !         &
        !                                                                 &
                CUSTOMER.NUMBER$        =   6                             &
        ,       CUSTOMER.NAME$          =  30                             &
        ,       ADDRESS.1$              =  30                             &
        ,       ADDRESS.2$              =  30                             &
        ,       ADDRESS.3$              =  30                             &
        ,       ZIP.CODE$               =   5                             &
        ,       TELEPHONE.NUMBER$       =  10                             &
        ,       ATTENTION.LINE$         =  20                             &
        ,       CREDIT.LIMIT$           =  12                             &
        ,       CURRENT.BALANCE$        =  12                             &
        ,       YTD.PURCHASES$          =  12                             &
        ,       NEXT.ORDER.NUMBER$      =   4                             &
        ,       NEXT.PAYMENT.NUMBER$    =   4                             &
        !                                                                 &
        \ MAP   (CUSTOM)                        ! Level = 05    !         &
        !                                                                 &
                FILLER$                 = 161                             &
        ,       ACCT.FILLER$            =   6                             &
        ,       ACCT.NUMBER$            =   6                             &
```

Figure B-6 The BASIC PLUS-2 TST FIND (3 of 8)

```
1000    !$*****************************************************************&
        !                                                                 &
        !                                                                 &
        !               M A I N    T S T    L O G I C                     &
        !                                                                 &
        !                                                                 &
        !*****************************************************************&
        !                                                                 &
        \ ON ERROR GO TO 19000                                            &
        !                                                                 &
        !       Set up standard default error trap                        &

2000    !$*****************************************************************&
        !                                                                 &
        !                                                                 &
        !               O P E N   D A T A   F I L E S                     &
        !                                                                 &
        !                                                                 &
        !*****************************************************************&

2200    !*****************************************************************&
        !                                                                 &
        !                                                                 &
        !               O P E N   F I L E   CUSTOM.DAT                    &
        !                                                                 &
        !                                                                 &

2210    \ CUSTOM.CHNL% = 3%                                               &
        !                                                                 &
        !       Assign the channel number.                                &

2220    \ OPEN "CUSTOM" AS FILE CUSTOM.CHNL%,                             &
               ORGANIZATION             INDEXED                          &
                                        VARIABLE,                        &
               ACCESS                   READ,                            &
               ALLOW                    READ,                            &
               MAP                      CUSTOM,                          &
               PRIMARY KEY              CUSTOMER.NUMBER$,                &
               ALTERNATE KEY            CUSTOMER.NAME$                   &
                                        DUPLICATES                       &
                                        CHANGES                          &
        !                                                                 &
        !       Open the Customer File.                                   &
```

Figure B-6 The BASIC PLUS-2 TST FIND (4 of 8)

```
3000    !$*****************************************************************&
        !                                                                    &
        !                                                                    &
        !       D E T E R M I N E   W H I C H   C A S E                      &
        !                                                                    &
        !                                                                    &

3010    \ GO TO 4100     IF EM.CUSTOMER.NUMBER$ <> "000000"                 &
        \ GO TO 4400     IF EM.CUSTOMER.NAME$ = "  "                        &
        \ GO TO 4200                                                        &
        !                                                                    &
        !       Three cases possible:                                        &
        !               Customer number entered,                             &
        !               Nothing entered (operator error),                    &
        !               Name entered,                                        &

4100    !$*****************************************************************&
        !                                                                    &
        !                                                                    &
        !       C U S T O M E R   N U M B E R   E N T E R E D                &
        !                                                                    &
        !                                                                    &

4110      GET #CUSTOM.CHNL%, KEY #0 GE EM.CUSTOMER.NUMBER$                   &
        \ WS.CUSTOMER.FILE.RECORD$ = CUSTOMER.FILE.RECORD$                   &
        \ GO TO 4800                                                        &
        !                                                                    &
        !       Otherwise, read the customer file.  ("Record not found"      &
        !         is trapped at statement 19020.)                            &
        !       Move the record into workspace,                              &

4200    !$*****************************************************************&
        !                                                                    &
        !                                                                    &
        !         C U S T O M E R   N A M E   E N T E R E D                  &
        !                                                                    &
        !                                                                    &
        !                                                                    &

4210      GET  #CUSTOM.CHNL%, KEY #1 GE EM.CUSTOMER.NAME$                    &
        \ WS.CUSTOMER.FILE.RECORD$ = CUSTOMER.FILE.RECORD$                   &
        \ GO TO 4800                                                        &
        !                                                                    &
        !       Set file pointer to first customer record for this           &
        !         customer name,                                             &
        !       Get the first record ("Record not found" is trapped          &
        !         at statement 19020.)                                       &
        !       Move record to workspace,                                    &

4400    !$*****************************************************************&
        !                                                                    &
        !                                                                    &
        !       N O   D A T A   F R O M   O P E R A T O R                    &
        !                                                                    &
        !                                                                    &
```

Figure B-6 The BASIC PLUS-2 TST FIND (5 of 8)

```
4410      REPLY.BUF$ = "Operator Error - No Data Supplied  "            &
        \ CALL REPLY BY REF (REPLY.BUF$,LEN(REPLY.BUF$),2%,STATUS%())   &
        \ GO TO 19950   IF STATUS%(0%) <> 1%                            &
        \ GO TO 32000                                                   &
        !                                                               &
        !       Restart Transaction with Reply Message                  &
        !       Abort transaction if bad status.                        &
        !       Exit TST.                                               &

4800      ED.RESULT$  =  FN.FORMAT$(EDIT.STRG$,WS.CREDIT.LIMIT$)        &
        \ WS.CREDIT.LIMIT$ = ED.RESULT$                                 &
        \ ED.RESULT$  = FN.FORMAT$(EDIT.STRG$,WS.CURRENT.BALANCE$)      &
        \ WS.CURRENT.BALANCE$ = ED.RESULT$                              &
        \ ED.RESULT$  =  FN.FORMAT$(EDIT.STRG$,WS.PURCHASES.YTD$)       &
        \ WS.PURCHASES.YTD$ = ED.RESULT$                                &

5000      CALL PRCEED BY REF (WS.CUSTOMER.FILE.RECORD$,205%,STATUS%())  &
        \ GO TO 19950 IF STATUS%(0%) <> 1%                              &
        \ GO TO 32000                                                   &

5200    !$**************************************************************&
        !                                                               &
        !                                                               &
        !       C U S T O M E R   I D   N O T   O N   F I L E           &
        !                                                               &
        !                                                               &

5210      REPLY.BUF$ = "No Record under that ID"                        &
        \ CALL REPLY BY REF (REPLY.BUF$,LEN(REPLY.BUF$),2%,STATUS%())   &
        \ GO TO 19950   IF STATUS%(0%) <> 1%                            &
        \ GO TO 32000                                                   &
        !                                                               &
        !       Send Reply if Record Not Found                          &
        !       Abort Transaction if Bad Status Return                  &
        !       Exit TST.                                               &
        !                                                               &
        !***************************************************************&

15000   !$**************************************************************&
        !                                                               &
        !                                                               &
        !       F U N C T I O N S     L O C A L     T O                 &
        !                                                               &
        !                       T H I S    T S T                        &
        !                                                               &
        !                                                               &
        !***************************************************************&
```

Figure B-6 The BASIC PLUS-2 TST FIND (6 of 8)

```
15001   DEF FN.FORMAT$( FIELD$, INPUT$ )
15010   NUMBER.FORMAT$ = TRM$( INPUT$ )                                    &
        \ NUMBER.FORMAT$ = STRING$(LEN(FIELD$)-LEN(NUMBER.FORMAT$),48%)    &
                         + NUMBER.FORMAT$                                  &
        \ FOR I.FN% = LEN(FIELD$) TO 1% STEP -1%                           &
          \ CHAR.FORMAT$ = MID( FIELD$, I.FN%, 1% )                        &
          \ GO TO 15020   IF (CHAR.FORMAT$ = '9')                          &
                          OR (CHAR.FORMAT$ = 'Z')                          &
          \ CHAR.FORMAT$ = SPACE$(1%)     IF CHAR.FORMAT$ = 'B'            &
          \ NUMBER.FORMAT$ = MID( NUMBER.FORMAT$, 2%, I.FN%-1% )           &
                 + CHAR.FORMAT$                                            &
                 + RIGHT( NUMBER.FORMAT$, I.FN%+1% )
15020      NEXT I.FN%                                                      &
        \ FOR I.FN% = 1% TO LEN(FIELD$)                                    &
          \ CHAR.FORMAT$ = MID( FIELD$, I.FN%, 1% )                        &
          \ CHAR1.FORMAT$ = MID( NUMBER.FORMAT$, I.FN%, 1% )               &
          \ GO TO 15030   IF CHAR.FORMAT$ = '$'                            &
                          OR CHAR.FORMAT$ = ','                            &
                          OR CHAR.FORMAT$ = 'B'                            &
          \ GO TO 15040   IF CHAR.FORMAT$ = '9'                            &
          \ IF CHAR1.FORMAT$ <> '0'                                        &
            AND CHAR1.FORMAT$ <> ','                                       &
            THEN  GO TO 15040                                              &
            ELSE  NUMBER.FORMAT$ =                                         &
                          SPACE$(1%)                                       &
                          + LEFT( NUMBER.FORMAT$, I.FN%-1% )               &
                          + RIGHT( NUMBER.FORMAT$, I.FN%+1% )
15030      NEXT I.FN%
15040    FN.FORMAT$ = NUMBER.FORMAT$
15050   FNEND                                                              &

19000   !$******************************************************************&
        !                                                                  &
        !                                                                  &
        !       S T A N D A R D     E R R O R     H A N D L I N G          &
        !                                                                  &
        !                                                                  &
        !******************************************************************&

19020   \ IF ERR = 155%                                                    &
          AND (ERL = 4110% OR ERL = 4210%)                                 &
          THEN                                                             &
            RESUME 5200                                                    &
        !                                                                  &
        !       Trap for customer id not on file.                          &

19040   \ IF ERR = 172% THEN RESUME 4800                                   &
        !                                                                  &
        !       Trap for Record Returned but locked                        &
```

Figure B-6 The BASIC PLUS-2 TST FIND (7 of 8)

```
14050   \ IF ERR = 154%                                                    &
          THEN                                                             &
          REPLY.BUF$ = "Access Denied, Record Locked by Another Task"      &
        \ CALL REPLY BY REF (REPLY.BUF$,LEN(REPLY.BUF$),2%,STATUS%())      &

19900   \ REPLY.BUF$ = " I-O Error Number "                          &
                       +NUM$(ERR)                                    &
                       +"at Line # "                                 &
                       +NUM$(ERL)                                    &
        !                                                            &
        !       For unexpected errors, go to                         &
        !         system default error dump out.                     &

14950   !$*************************************************************&
        !                                                            &
        !                                                            &
        !       A B O R T   T H E   T R A N S A C T I O N            &
        !                                                            &
        !                                                            &
        !*************************************************************&

14960     CALL ABORT BY REF (REPLY.BUF$,LEN(REPLY.BUF$),2%,STATUS%())    &
        \ GO TO 32000                                                    &
        !                                                                &
        !  Standard ABORT handling:                                      &
        !       Send Reply with Abort to Terminal Station                &
        !       Call TSTLIB routine to abort transaction.                &
        !       No return is expected but nevertheless provide           &
        !         an orderly exit from TST.                              &

32000   !$*************************************************************&
        !                                                            &
        !                                                            &
        !       E N D    O F    P R O C E S S I N G                  &
        !                                                            &
        !                                                            &
        !*************************************************************&

32767   !$*************************************************************&
        !                                                            &
        !                                                            &
        !               E N D    O F    T S T                        &
        !                                                            &
        !                                                            &
        !*************************************************************&
        !                                                            &
        \ TSTEND
```

Figure B-6 The BASIC PLUS-2 TST FIND (8 of 8)

Figure B-7 Form Two of the Display Transaction

APPENDIX C

GLOSSARY OF TERMS

aborting a transaction Terminating a transaction instance before successful completion.

application A logically related set of data processing operations which support a particular business activity.

application data file A file which supports an application's data storage requirements. Applications have two kinds of data files: permanent files, which support the application's ongoing data storage requirements, and work files, which provide transient data storage.

application program A program written for a particular application. TRAX supports two kinds of applications: Transaction Step Tasks, which are components of transaction processors; and stand-alone application programs, which execute in the support environment.

application terminal A TRAX terminal reserved for use by a transaction processor.

Application terminals can not be connected to the same PDP-11 interface device as support terminals, but application terminals used by different transaction processors can share a common interface device.

Application Terminal Language (ATL) The language used to create form definitions for transaction processors. Using ATL, one can specify: the layout and appearance of each form; the attributes of each field on those forms; the format of the exchange message assembled from the user's input; and the set of replies with which the transaction processor can respond to the user's input.

ATL (1) Application Terminal Language, (2) the TRAX utility program which manages Forms Definition Files.

authorization See terminal authorization, user authorization.

batch control command A batch processor directive placed in a batch control file. TRAX batch processors support the following six batch control commands in addition to the DCL commands accepted at support terminals: $JOB, $DATA, $EOD, $IF, $ON, and $EOJ.

batch control file A file containing sufficient support terminal dialogue and batch control commands to control a batch processor.

batch processor A TRAX system component which executes a series of system commands and support programs as directed by batch control files.

By submitting a batch control file to a batch processor, system commands and support programs can be executed without operator intervention and without dialogue at a support terminal.

batch stream The processing of a batch control file by a batch processor.

caching Storing a duplicate copy of data into a temporary storage area that can be more quickly accessed than that data's primary storage area. Caching is done to speed access to frequently used data.

data file access Data file access is provided in two ways, depending upon the nature of the file being accessed.

Permanent files are automatically opened when the transaction processor is installed, and are kept open while the transaction processor is in operation. These files are shared among all TSTs in the transaction processor, and multiple TSTs can update the same file simultaneously. When referring to these files, TSTs use the logical file names specified in the transaction processor generation procedure.

Work files are created, opened, and closed only when directed by a TST. Each create operation produces a file with a unique file specification. A TST can obtain the unique file specification via a library call. These files can not be shared between TSTs, and only one TST can have update access to a given work file at a time.

deadlock A situation in which each of several transaction instances has successfully acquired a resource, and in which at least two of the instances simultaneously need access to a resource acquired by the other in order to proceed.

DIGITAL Command Language (DCL) The command language used in the support environment. DCL can be used for support terminal dialogue and in batch control files.

entry form A form used in a situation where a user response is expected. A transaction instance can request the display of an entry form only on the terminal which initiated that transaction instance.

exchange A cycle in the processing of a transaction instance consisting of the processing of an exchange message by one or more stations.

An exchange commences with the placement of the exchange message (containing the data to be processed) into the transaction slot. It ends when the exchange message is discarded following the termination of the last transaction step in the message's routing list.

exchange message A station message that resides in a transaction slot and accompanies the transaction slot for an entire exchange. Each station in the exchange route can read and/or modify the contents of the exchange message. The routing list of the exchange message determines the processing of the associated transaction slot.

exchange recovery The automatic restart of an exchange. Only the processing for the affected exchange is restarted; the data which began the exchange is not recollected and the same exchange message is reused. All of the staged file updates done by the previous unsuccessful processing are automatically removed before the restart is attempted.

Repeated exchange recovery failures will cause the transaction instance to time out and therefore to be aborted.

file specification A complete RMS-11 file identifier. This identifier consists of device name, account specifier, file name, file extension, and version number.

File specifications must be used by all support programs when referring to work files. When TSTs refer to application data files (permanent or work), they must use logical file names.

form The information structure used to collect or display data at application terminals.

form definition A detailed specification of a form. Form definitions specify the layout of a form; the rules by which a user must fill out the form; the format of the station message that will supply initial field values for the form; the format of the exchange message that will be built from the user's input; and the format of each of the replies which the transaction processor can use with the form.

The specifications for all forms used by a transaction processor are stored in a forms definition file. Forms definitions are written in ATL and placed in the forms definition file by the ATL utility program.

forms definition file A file associated with a transaction processor containing definitions for all forms used by that transaction processor.

forms definition record A record in a forms definition file. Each forms definition record contains the coded definition of a single form.

journaling Recording updated data file records on an alternate medium to protect against failure of the primary medium.

If such a failure should occur, a backup version of the primary medium together with the updated records on the alternate medium can be used to recreate the state of the primary medium at the time of the failure.

TRAX transaction processors achieve journaling by writing the transaction slot to the journal medium at the conclusion of any transaction instance which updates a file requiring journaling. If journalled, a file will also be staged so that the updated records will be present in the transaction slot. Because journaling is postponed until the conclusion of a transaction instance, an aborted transaction will not cause a journal entry.

Journaling is specified on a file-by-file basis during transaction processor generation.

link A logical data transmission path between two transaction processors consisting of a master link station in one transaction processor, one or more slave link stations in another transaction processor or in an IBM system, and a means of data transfer between them. See also sub-link.

logical file name A name of up to six characters used within a transaction processor to identify an application data file.

The transaction processor generation procedure associates these logical names with complete file specifications.

logging Recording station messages and other TST-specified data onto the journal media.

mailbox message A station message sent from a TST to a mailbox station. The message data is stored on a disk file by the receiving mailbox station.

mailbox station A station to which mailbox messages are sent to await collection by a TST.

master link station A link station to which exchange messages are routed to cause either the initiation of a transaction instance within another transaction processor, or the transfer of additional data to such a transaction instance.

partition A segment of memory allocated to one or more tasks.

priority See station priority.

record lock The restriction of access to a record.

When a transaction instance requires a record for updating, it locks that record so that no other transaction instance can update it. (The record can remain available, if desired, for read-only access by other transaction instances.) When the original transaction instance performs an unstaged update, releases an unstaged lock, or finishes all staged file operations, the lock is removed and the record again becomes available to other transaction instances for updating.

All locks are exercised on behalf of the transaction instance, rather than on behalf of the transaction step which issued the lock request. Locks therefore survive the termination of individual transaction steps.

Transaction instances which request update access to a locked record are temporarily suspended, and the access is retried automatically after a pre-defined delay. If the record is still locked at that time, either the current exchange for the transaction instance will be restarted, or the entire transaction instance aborted. The designer makes this selection by his choice of exchange recovery for the corresponding transaction definition.

report form A form used in a situation where no user response is possible. A request (REPORT MESSAGE) for the display of a report form can be directed only to an output-only terminal station.

response message A station message sent from a TST processing the current exchange message back to the source station for the transaction instance. Response messages are used to return display data, exchange control, and form control to the source station.

RMS-11 The PDP-11 file management facility through which TRAX tasks access data files.

RMS-11 is available to TRAX programs in two modes. TSTs which are part of a transaction processor will perform all file operations through the TRAX executive service which contains complete RMS-11 file support, and TSTs can therefore make full use of standard RMS-11 facilities without having to directly include RMS-11 support.

Support programs must directly include RMS-11 file support during the linking process.

RMS-11 provides facilities to support sequential, relative, and indexed files in combinations of fixed and variable record lengths.

routing list A list of station names to which an exchange message is to be sent.

slave batch station A station which can initiate a single-exchange transaction upon request from a support environment program.

slave link station A station which can initiate a transaction instance upon request of another transaction processor via a sub-link.

spooler A TRAX utility program which processes queued file printing requests.

staging The postponement of updates to a file until successful completion of a transaction instance. Aborted transactions have no effect on staged files, because no updates will have occurred.

station A logical location within a transaction processor where station messages are received and processed. Each TRAX transaction processor has a set of stations, each station having a unique name of up to six alphanumeric characters. Stations are classified as terminal, Transaction Step Task (TST), master link, slave link, mailbox, slave batch, or submit batch.

station message Formatted data sent to one or more transaction processor stations. There are four types of station messages, EXCHANGE, RESPONSE, REPORT, and MAILBOX.

station priority An attribute of a TST station which determines the startup priority of its associated Transaction Step Task.

submit batch station A station to which exchange messages are sent to cause the submission of batch command files to a batch processor.

sub-link A subdivision of a link consisting of the logical data path between the master link station and a single slave link station. A link can have one or more sub-links, the number of sub-links being determined by the number of slave link stations associated with the link.

support environment The facilities which TRAX provides for the execution of support programs. The support environment is used for the following kinds of processing: control and supervision of transaction processors, batch processors, and spoolers; execution of application-specific support programs; editing, compilation, task building, and debugging of programs; generation of transaction processors; file backup and recovery operations.

None of the facilities described for the transaction processors are available to programs running in the support environment. Such programs can not update files to which a transaction processor has write access.

support program A program which is not part of a transaction processor. Support programs run under the control of a support terminal or under a batch processor.

support terminal A TRAX terminal which is used for the execution of system commands and support programs. Support terminals can not be connected to the same PDP-11 interface device as application terminals.

terminal See application terminal and support terminal.

terminal authorization The association of a set of transaction definitions with an application terminal. Only those transactions included in the set can be performed at such a terminal while the terminal authorization remains in effect.

An application terminal's initial terminal authorization can be selected by supplying the name of a work class when the corresponding terminal station is defined. Terminal authorizations can be overridden by user authorizations, and are disregarded by those transaction selection forms which do not check authorizations.

terminal station The transaction processor component which controls an application terminal.

The terminal station is responsible for the display of forms, the receipt of user responses to forms, the construction of exchange messages from these responses, the initiation of exchanges using the received data, and the display of any replies requested during exchange processing.

There are two types of terminal stations: 1) Interactive terminal stations are associated with application terminals which can initiate transaction instances. Each transaction instance initiated by an application terminal can direct display requests to the corresponding terminal station. A transaction instance cannot direct display requests to any interactive terminal station other than the one associated with its initiating terminal. 2) Output-only terminal stations are associated with application terminals which are used in an output-only mode. Any transaction instance can direct display requests to an output-only terminal station. These requests will be processed on a first-in, first-out basis at each station.

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____
or
Country

---------- Fold Here ----------

---------- Do Not Tear - Fold Here and Staple ----------

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Documentation
146 Main Street ML 5-5/E39
Maynard, Massachusetts 01754