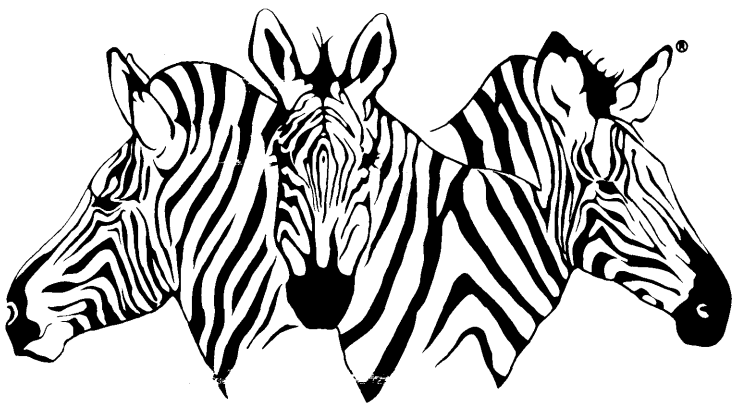
overview of the PICK operating system

88A00751A03



ZEBRA® BY GA

RECORD OF REVISIONS

Title: Overview of the PICK Operating System

Document No. 88A00751A03

Date	Issue
Nov 82	Original Issue
Sep 83	Revision B
Mar 84	Revision C - A03

NOTICE

The information contained in this document is subject to change without notice.

General Automation makes no warranty or representation with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. General Automation shall not be liable for errors contained herein.

General Automation assumes no responsibility for the use or reliability of its software on equipment that is not furnished by General Automation.

overview of the PICK operating system

88A00751A03

1045 South East Street P.O. Box 4863
Anaheim, California 92803
(714)778-4800 (800)854-8334
TWX 910-591-1695 TELEX 685-513

RECORD OF REVISIONS

Title: Overview of the PICK Operating System

Document No. 88A00751A03

Date	Issue
Nov 82	Original Issue
Sep 83	Revision B
Mar 84	Revision C - A03

NOTICE

The information contained in this document is subject to change without notice.

General Automation makes no warranty or representation with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. General Automation shall not be liable for errors contained herein.

General Automation assumes no responsibility for the use or reliability of its software on equipment that is not furnished by General Automation.

FOREWORD

This manual has been created for the individual who is not familiar with the PICK Operating System, which is part of the General Automation ZEBRA™ series. The intent is to provide as much information as possible covering the capabilities of the system, in general, and the data base management facility, in particular. Because of this approach, the manual has a different flavor that should be explained.

First, it is not a "working" manual in the sense that one can sit down and actually program or otherwise use the computer with the manual as a tutorial guide. In almost every case, the syntax and many details of a given function have been omitted, and many functions have been glossed over, or left out altogether. The intent here is to provide, in brief, a feel for the capabilities of the system, not to explain precisely how to use it. For the latter, the reader is referred to the PICK Operator Guide and the PICK processor reference manuals listed below.

This manual is slanted towards the unusual systems software and data base capabilities, rather than the hardware and more conventional software functions that are a part of most computers. There is much about the PICK Operating System that is truly different and unique; the sections on the BASIC language, and certain other functions, have been dealt with briefly with the assumption that most readers will already be familiar with the general principles.

On the other hand, the sections on file structure which then lead into the section "Data Base Architecture and the Access Process" digress into great detail. The reasoning here is that the system is quite different, and a basic understanding of data base principles is necessary before one can appreciate the specific implementation. These first sections are also the key to understanding the rest of the system, and the reader is encouraged to concentrate on these first before attempting to read further.

Other ZEBRA documentation that is available from General Automation:

Document No.	Title
88A00757A	PICK Operator Guide
88A00758A	ACCU-PLOT Operator Guide
88A00759A	COMPU-SHEET Operator Guide
88A00760A	Quick Guide for the PICK Operating System
88A00774A	PICK Utilities Guide
88A00776A	PICK ACCESS Reference Manual
88A00777A	PICK SPOOLER Reference Manual
88A00778A	PICK BASIC Reference Manual
88A00779A	PICK EDITOR Reference Manual
88A00780A	PICK PROC Reference Manual
88A00781A	PICK RUNOFF Reference Manual
88A00782A	Introduction to PICK TCL and FILE STRUCTURE
88A00783A	PICK JET Word Processor Guide

TABLE OF CONTENTS

1 introduction

The PICK Operating System is a new kind of operating system, principally oriented towards data base management applications. Computer users are discovering, sometimes belatedly, that many applications, reduced to their essence, are data base management tasks. Applications which use traditional file structures are usually thought of as being "optimized," yet management decision makers are finding that lack of flexibility in a data base is a far from "optimum" condition, and is rapidly becoming regarded as an intolerable misuse of a valuable asset. The decreasing cost of computational logic and data storage, and the increasing awareness that a well-managed data base can become a valuable decision making tool, is the driving force behind data base management systems today. In this light, to say the PICK Operating System is a data base management operating system is not to limit its application scope, or to imply that it is somehow a special purpose operating system. Rather, it is an operating system whose time has come.

1.1 VIRTUAL MEMORY OPERATING SYSTEM

The PICK Operating System "addresses" disk storage in 512 byte units called "frames." This technique is similar to "paging" in virtual memory management systems commonly used in very large scale mainframe computers. Frames are moved in and out of semiconductor memory dynamically, on an as-needed basis. This process is totally transparent to the individual user, who, for all practical purposes, can manipulate data within the capacity range of the given disk configuration.

1.2 DICTIONARY-BASED FILE STRUCTURE

A given file is defined by entries in a "file dictionary." A file is subdivided into open ended "groups" and a hashing algorithm takes a keyword and computes the group in which an individual "item" will be stored. On retrieval, only one group, typically a very small sub-set of the entire file, must be string searched to locate the item.

1.3 RETRIEVAL LANGUAGE

The PICK Operating System includes a high-level, non-procedural relational data base query facility ("ACCESS"). This flexible software tool can be used by non-technical personnel to make simple ad hoc inquiries, yet has powerful features that can be used to formulate very complex reports in far less time than would ordinarily be the case using conventional programming languages. This translates directly into lower data processing costs, easier customization, and modification of any given software, and faster development of new applications.

1.4 SUMMARY

The PICK Operating System represents a different approach to computing, and provides a wide variety of fully-developed software products to implement that approach. Most data processing tasks can benefit from the flexibility of the PICK Operating System file structure and data base management tools, and few systems offer the cost/performance of such sophisticated operating software on a quality mini or micro computer configuration.

file structure 2

2.1 INTRODUCTION

The file management system that is central to the operation of the PICK Operating System represents an unusual approach to the use of disk storage. Using this system, the operating system can find the location of a given "item" with a minimum number of disk seeks. Data within this item can be further subdivided with special delimiter marks the system is optimized to search for. The English-like ACCESS Processor is designed to make these operations as simple, and as user oriented, as possible.

2.2 BASIC DATA DIVISION CONCEPTS

This section discusses the system file structure mechanics on a detailed level. Certain basic concepts are briefly introduced here in order to make the information in remaining sections more easily understood. Each of these concepts are individually discussed in greater detail.

2.2.1 ITEMS

A PICK Operating System file is made up of "items." Items consist of a keyword called the item-id that uniquely identifies the item from all other items in the file. An item can consist of up to 32K of data, divided into a number of "fields" called attributes, which can be further subdivided.

2.2.2 DICTIONARIES

Files have "file dictionaries" associated with them. These dictionaries contain, among other things, the location on disk of the related file.

2.2.3 GROUPS

Files are divided into one or more "groups." The item-id (keyword) of each item in the file is run through a hashing algorithm to determine which group that item will be stored in. Upon retrieval, only that group need be searched.

2.3 DISK STORAGE ALLOCATION

Disk storage is divided into 512 byte "frames." Each one of these frames has a logical address known as the "frame-id" and each frame contains twelve bytes of information the system uses, including the number and location of forward and backward "links" in the chain of frames that make up an individual file. Five hundred bytes per frame are available for user data storage.

When a new file is created, space for it on disk is reserved in one contiguous set of frames. These frames are chosen by consulting the available space pool using two figures supplied by the user, the "modulo" and the "separation." This space is called the "primary" space, and in no way represents a limit on how large the file can grow.

2.4 THE MODULO AND THE SEPARATION

The modulo and the separation are a method of dividing up a whole file into smaller groups. The purpose of dividing a file into groups is to focus a search for a given item of data in a smaller amount of storage, and thus minimize the search time. The modulo represents the number of groups the file is to be divided into. The separation represents the number of frames that will initially be allocated to each group. If a file is "modulo three," that means the file is divided into three groups. If the separation is three, then each group is three frames long, and the file would reserve a total of nine frames as the "primary" filespace.

In short, the modulo*separation represents the total number of frames allocated for the file and these frames are always contiguous.

It is important to emphasize that this pre-allocation places no limit on the growth potential in a file. Because each frame has forward and backward linking pointers, new frames are automatically added to an expanding group, as required.

While using the modulo*separation to determine the amount of contiguous space to allocate to a file, the system also uses these figures to build the file dictionary. The file dictionary will contain the modulo, the separation, and the disk address of the first frame in the file, called the "base frame" or "filebase."

The conceptual effect of dividing a file into groups is to create a number of separate files. Figure 2-1 diagrams the "primary" space of the modulo-3 separation-3 file mentioned earlier. This file is divided into three frames each. The "base frame" is the first frame of the reserved, contiguous "primary" filespace.

Notice the file effectively contains three "starting points" in frames 100, 103, and 106. These frames all "link backward" to frame 0, which is to say they do not "link back" at all. Because only the first frame of Group 0 is stored in the file dictionary as the "base frame" or "filebase," the starting frames of the following groups are computed. Since the "primary space" frames are always contiguous, the base frame of any given group can easily be determined using the following formula:

Base of any Group = (Group * Separation) + Filebase

Thus, the starting frame of Group 1 is 103:

103 = (1 * 3) + 100

Now suppose an item is to be added to the file. First the item-id is hashed to determine which group it will be stored in. The hashing algorithm will return a value within the range of the modulo (the number of groups in the file). In this case, we will assume the result is Group 1. Also, we will assume that the primary space allocated to Group 1 is almost full and the addition of this item will cause it to overflow two frames. Figure 2-2 diagrams the result.

As Figure 2-2 shows, when Group 1 overflowed, the additional data was written into frames 947 and 1106 and the linking information was used to chain these frames to the last frame of the "primary" space in Group 1. Should the total size of Group 1 be reduced by future deletions, frames 947 and 1106 would be "un-linked" and returned to the free space pool. The "primary space" will always be associated with the file, even if one group, or the entire file, has all items deleted.

This process of defining the modulo and separation for a file can be used to optimize the file access process. Utility programs are available to choose the optimum modulo and separation based on average item size and the total number of items stored. Since the search for any one item is always limited to only one group, it is group size, not file size, that determines the speed of retrieval. The separation, involving reservation of contiguous space, is an attempt to ensure that the "next access" will be physically the next sector on the disk. Since most PICK Operating Systems are used in a timesharing environment, contention for the disk will almost always cause head movement before the next access for any one user. Therefore, in most instances, the separation is left as one, and the linking process begins as soon as the first frame of a group is filled; each group being free to grow and shrink independently.

MODULO	3
SEPARATION	3
BASE FRAME	100

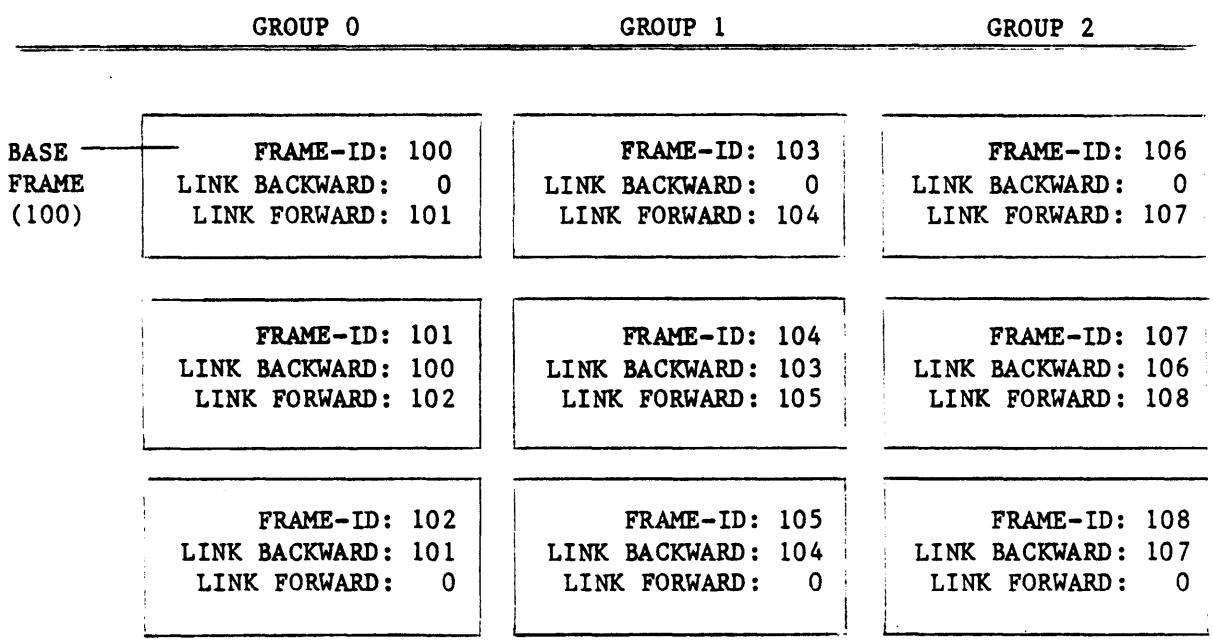


Figure 2-1

MODULO	3
SEPARATION	3
BASE FRAME	100

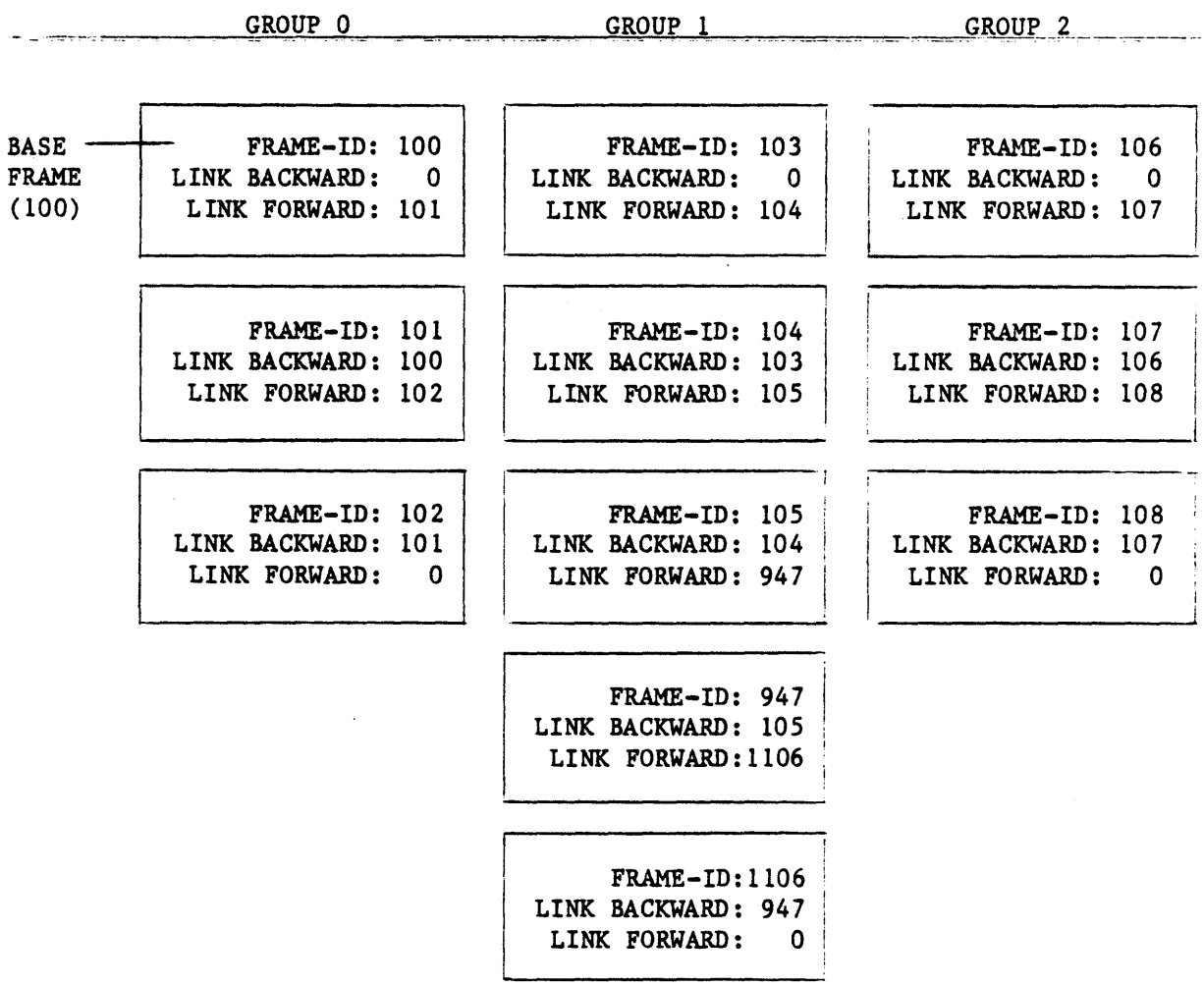


Figure 2-2

2.5 NATURE OF AN ITEM

A "group" is a more or less conceptual division of a file into subdivided space. After the proper modulo and separation is chosen, the fact that a file is divided into groups becomes transparent to the user. The item is where the real mechanics of the PICK Operating System file structure begin.

An item-id is a keyword that identifies a related group of fields called an item. A file is a collection of related items. A file can be any length, consisting of any number of items, but any one item is limited to 32K bytes. An item is divided into attributes, and these attributes can contain any data except for attribute 0, the 0th attribute, which contains the keyword that identifies the item. This attribute 0, called the item-id, can be up to 50 bytes long. No other length restrictions apply to individual attributes other than their collective limit of 32K bytes.

The data that follows the item-id, "attribute 0," can be further divided into more attributes; attributes can contain multiple values; and values can contain multiple subvalues. Attributes are delimited by physically writing the ^ character to disk between attributes. Multiple values within attributes are delimited by the] character and multiple subvalues within values are delimited by the \ character. Attributes, values, and subvalues are individually variable in length, can grow or shrink as required by the application, and occupy only as much disk storage as they require plus the one-byte delimiter marks that separate them. All of the information pertaining to what attributes are in a file is contained in the file dictionary. Whereas attributes, values, and subvalues can potentially contain the same data, attribute 0, the item-id, must be unique within one file.

The special significance of attribute 0, or the item-id, is the further refinements used to locate it. The item-id is a keyword referring to up to 32K bytes of data that follows it, and it is limited to 50 or less alphanumeric characters which are computed in a hashing algorithm along with the modulo. The result of this computation is the number of the group in which the item is stored.

The retrieval of a given item follows this basic procedure:

1. Given the file name and item-id of the data to be retrieved, the operating system consults the file dictionary and determines the starting frame, modulo, and separation of the file.

2. Using the item-id supplied by the user and the hashing algorithm, the starting frame of the group in which the item must be stored is determined.

3. The starting frame and associated linking pointers to the following frames are read and the group is searched until the item-id is found.

To illustrate the format of an individual item, Figure 2-3 shows a hypothetical item in an inventory file.

Figure 2-3 illustrates the role of the attribute mark ^, the value mark], and the subvalue mark \. Both attributes 1 and 2 contain one item of data or "value." Attribute 3 illustrates the full range of complexity that can be managed by the operating system, it contains three values delimited by the value mark], and the third of these, Value 3, consists of two subvalues.

Any attribute can contain multiple values, and any value can contain multiple subvalues. In Figure 2-3, attribute 1 could be text listing a vendor name, attribute 2 could be the vendor address, and attribute 3 could be multiple parts supplied by the vendor. Subvalues of Value 3 of attribute 3 might list color options for an individual part, if any. The following list graphically illustrates the relationships between multiple values in this type of file:

```
Attribute 1 ----> VENDOR NAME
Attribute 2 ----> VENDOR ADDRESS
Attribute 3, Multivalues ----> 1  PART A
                               2  PART B
                               3  PART C
Attribute 3, Subvalues ----------------------> 3-1  SAFETY ORANGE
                                               3-2  FLAT BLACK
```

```
PART-52900^DATA FIELD #1^DATA FIELD #2^VAL #1]VAL #2]VAL #3-1 VAL #3-2
          |             |              |                              |
          +Attribute 1--+--Attribute 2--+-------+--Attribute 3--------+
                                        |       |      |
                                        |       |      +--Subvalues 3-1&3-2
                                        |       +--Value 2
                                        +--Value 1
```

Figure 2-3

The PICK Operating System BASIC, discussed in Section 9, includes many commands for accessing attributes, values, and subvalues.

When the data is displayed via the editor, the attribute marks are usually converted into carriage returns so the display is one of line entries. For example, a simple "copy to terminal" image of the previous example on a user's terminal might look like this:

```
    PART-52900
001 DATA FIELD #1
002 DATA FIELD #2
003 VAL #1]VAL #2]VAL #3-1\VAL #3-2
```

Notice that attribute 0, the item-id, is not given a line number. Although it is an attribute, it is reserved for its special purpose as a keyword. Because the very location of the data is dependent on the contents of attribute 0, it cannot be altered as a line item. Unlike attribute 0, the other line items have no particular significance to the file system, since their meanings are dependent upon human interpretation. Later we will show how entries in the file dictionary can give special meaning to these line items for interpretation by the ACCESS Processor and how that processor can be used to format more useful printouts than the above, somewhat mechanical, presentation.

2.6 SUMMARY

The PICK Operating System file structure provides a sophisticated system for the storage of data by keyword, attributes of the keyword, multiple values per attribute, and multiple subvalues per value. The storage technique assures that, given the keyword and certain data stored in the file dictionary, a given item can be located with a minimum number of disk seeks and string searching. Although the file structure uses disk space reservation techniques, there is no limit to the amount a file can grow. An individual item can also grow, regardless of the amount of space originally allocated to it, as long as the total item length does not exceed 32K bytes. Individual attributes, values, and subvalues within an item can also shrink and grow within the 32K limit.

dictionary system 3

3.1 INTRODUCTION

The PICK Operating System revolves around a hierarchy of files called "dictionaries." Dictionaries are used to describe the structure of other files and to "point" to their location by storing the base frame, modulo, and separation. This pointer information is central to the purpose of the dictionary. The existence of a file and its associated dictionary are integrally linked. Access to a file is not possible without a dictionary to regulate that access; and, where a single dictionary can serve several files, no single file can exist without a dictionary to define its location and structure.

In addition to storing the location of a file on disk, the information contained in the file dictionary serves as a roadmap for retrieving data from the associated data file when using the English-like ACCESS Processor. Entries give mnemonic names for various attributes, describe the contents of an attribute, the way an attribute is to be printed out, conversion specifications for dates and other entries, etc.

The hierarchy of the PICK Operating System dictionaries is as follows:

1. The System Dictionary contains the users who may log onto the system. There is only one system dictionary per system. An item-id (attribute 0) in the system dictionary is a user's name (or logon word); further attributes describe passwords and security codes, etc. The most important attributes of an item in the system dictionary are attributes 2, 3, and 4 which give the base frame, modulo, and separation of an individual account master dictionary.

2. The Master Dictionary is assigned when a new account is put on the system. Each account has only one master dictionary. The master dictionary, or "MD," contains entries which describe, or point to, the system command language a user has at his/her disposal. These items consist of system commands, called verbs; stored procedures, called PROCS; and commands of the ACCESS retrieval language. All of these elements will be discussed in further detail in subsequent sections. For now, the important point is that each user's command vocabulary can be customized, and sensitive commands can be omitted from any given account, effectively preventing use of that command. The MD also contains pointers to other files, called file dictionaries.

3. The File Dictionary contains file and attribute definition statements that describe the structure of the data file with which the dictionary is associated. These definition items can describe, on an attribute-by-attribute basis, the type of data in an attribute, conversion specifications, relationships between attributes, etc. The file dictionary can either point to a single file, or multiple files having identical structure. However, no file can be without a dictionary pointing to it. A special type of file, the "single-level" file, is a dictionary file with no associated data file. All of the data is contained within the dictionary itself, and it "points" to itself.

Figure 3-1 diagrams the hierarchy of dictionaries.

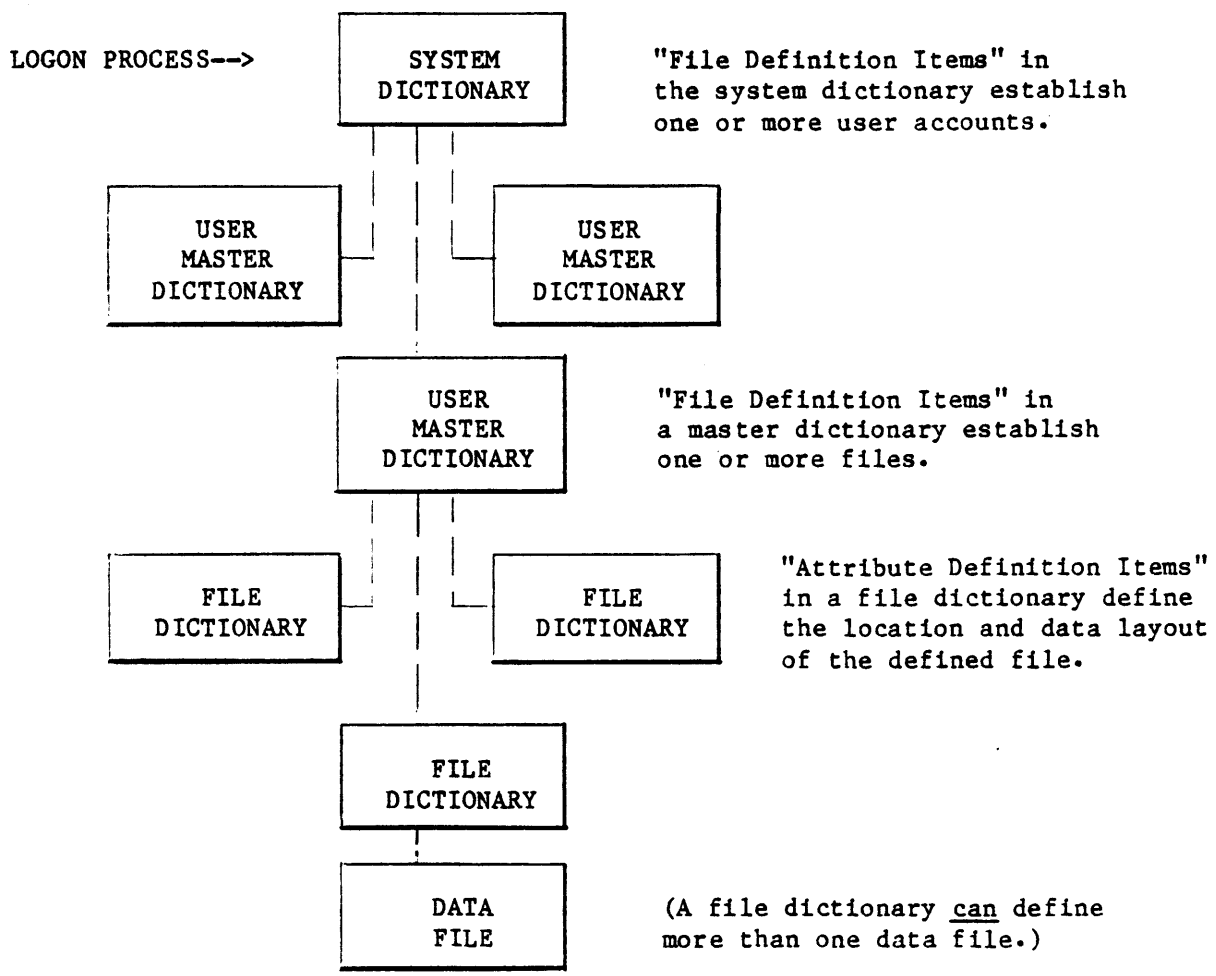


Figure 3-1. The Dictionary Structure

3.2 ITEMS AND ATTRIBUTES IN DICTIONARIES

As discussed earlier, all files consist of one or more items. Each is identified by a keyword, the item-id, that can be located quickly through the hashing algorithm. An entire item is, in reality, a string of attributes delimited by the character ^, the first one, attribute 0, serves as the key and functions as the item-id.

An important concept to remember is that dictionaries are files too. Their structure follows the same pattern of item-id followed by attributes, like any other file. Dictionaries achieve special significance by following a relatively rigid structure, unlike data files which follow whatever structure the data is suited to. Dictionary files reserve certain characters for attribute 1. If one of these reserved characters appear as attribute 1 in the item, then the item-id and the following attributes in that item take on special significance in the definition of the item and its purpose. There are several different classes of items that can appear in a dictionary file, the following is a discussion of the major types.

3.3 FILE DEFINITION ITEMS

When an item in a dictionary file is used to define another "lower level" file, the item-id, or attribute 0, of the item becomes the name of the file being pointed to. This lower level file can be either a true data file, or another dictionary.

When a file definition item appears in the system dictionary, the "file" being defined is the master dictionary of a user account. Some of the attributes take on special roles in the system dictionary such as the establishment of logon passwords.

When a file definition item appears in a master dictionary, the "file" being defined is the dictionary establishing a data file. One dictionary file can contain one or more file definition items pointing to data files that have identical structure.

3.3.1 ATTRIBUTE 0, THE ITEM-ID

The item-id becomes the name of the file being defined.

3.3.2 ATTRIBUTE 1

Within a dictionary file, an item that has the character D in attribute 1, also referred to as a D-ITEM, indicates that the following attributes define the structure of another file or dictionary. If the D in attribute 1 is followed by an X, the file will be erased during a system restore. This is useful when creating temporary files, rather than writing special purging routines. If the D is followed by a Y, the data within the file will be erased during a system restore, but the pointer to the file in its related dictionary will remain; thus the file still exists, it is just empty. If the D is followed by a C, the data within the file is binary data (compiler object code) and the file definition is actually stored in a pointer file, either the user's or the system pointer file (a pointer file is a file which contains special definition items that point to the disk location of object code files and selected lists of item-ids). DCX and DCY give these binary data files the X and Y purge control flags.

3.3.3 ATTRIBUTE 2

Attribute 2 contains the base frame-id of the file being defined. When the file is created and the contiguous space is reserved, the create file processor automatically loads this attribute; it, of course, must not be altered by the user.

3.3.4 ATTRIBUTE 3

Attribute 3 contains the modulo of the file being defined. When the file is created and the contiguous space is reserved, the create file processor automatically loads this attribute; it, of course, must not be altered by the user.

3.3.5 ATTRIBUTE 4

Attribute 4 contains the separation of the file being defined. When the file is created and the contiguous space is reserved, the create file processor automatically loads this attribute; it, of course, must not be altered by the user.

3.3.6 ATTRIBUTE 5

This is the retrieval lock code, a form of password protection discussed in Section 6, System Security.

3.3.7 ATTRIBUTE 6

This is the update lock code, a form of password protection discussed in Section 6, System Security.

3.3.8 ATTRIBUTE 7

When the file definition item is used to define a data file, this attribute is not used. When a file definition item in the system dictionary is used to create a user's master dictionary, this attribute contains the logon password.

3.3.9 ATTRIBUTE 8

When the file definition item is used to define a data file, this attribute is not used. When a file definition item in the system dictionary is used to create a user's master dictionary, this attribute contains the system privilege level of the defined user (refer to Section 6, System Security, for more information).

3.3.10 ATTRIBUTE 9

When the file definition item is used to define a data file, this attribute contains the type of data in the item-id field of the defined file. This includes alphabetical or numeric, and the output specifications for the line printer or terminal screen (flush left or right). The flush left/right option is taken into consideration when sorting the file by item-id.

When a file definition item in the system dictionary is used to create a user's master dictionary, this attribute contains the user-type code of the defined user (refer to Section 5, System Software and Utilities, for more information).

3.3.11 ATTRIBUTE 10

Attribute 10 designates the space to be allowed when printing the item-id.

3.3.12 ATTRIBUTE 11

Reserved for system use.

3.3.13 ATTRIBUTE 12

Reserved for system use.

3.3.14 ATTRIBUTE 13

Attribute 13 contains the new modulo and separation when the file is reallocated during the system restore process, called file-restore. If the modulo and/or separation of a file needs to be changed, the user puts the new values in this attribute. The next file-restore will examine these values and reallocate the file accordingly.

3.4 FILE-SYNONYM DEFINITION ITEMS

When an item in a dictionary file is used to define another "lower level" file, the item-id, or attribute 0, of the item becomes the name of the file being defined. It is sometimes convenient to give a file more than one name, as in giving the file "INVENTORY" the name "INV" for short. File synonyms can also be useful in a master dictionary, M/DICT, to alter the command language terminology and/or create abbreviations. The operating system makes this possible with the "file-synonym definition" item. Aside from giving a file an alternate name within the same user account, the file-synonym definition item can also point "outside" its account and reference files in other accounts, provided security restrictions are met. The item-id of the synonym definition item is the new version of the name. The "real" name is placed in attribute 3. The following is a summary of the attributes in file-synonym definition item.

3.4.1 ATTRIBUTE 0, THE ITEM-ID

The item-id becomes the synonym name of the file being defined. If the synonym item is being used to point to a file in another account, the item-id is the "real" name of the file.

3.4.2 ATTRIBUTE 1

Within a dictionary file, an item that has the character Q in attribute 1, also referred to as a Q-ITEM, indicates that the following attributes point to the name and account location of another file.

3.4.3 ATTRIBUTE 2

Attribute 2 contains the account name where the file "pointed to" by the Q-ITEM is located. This is also used to set up access to a file in another account. In this case, the same name for both the original file, the "D-ITEM," and the new synonym, the "Q-ITEM" might be the same. Thus the ability of a Q-ITEM to become a synonym is not used, only its ability to "point" to a file in another account. Access is still controlled by the security system. See Section 6, System Security, for a more detailed discussion. If attribute 2 is null, the file being pointed to is assumed to be in the same account.

3.4.4 ATTRIBUTE 3

Attribute 3 contains the name of the file as defined in the "D-ITEM." When referring to a file by its abbreviated "Q-ITEM" name, the system will search back to the original file definition item for the base frame, modulo, separation, and other attributes for purposes of file access.

3.5 ATTRIBUTE DEFINITION ITEMS

A data file, like all files, consists of items identified by the item-id followed by a number of other attributes, 1-n (or no attributes). These attributes make up the data in the file. The purpose of the attribute definition item in the file dictionary is to define the nature of the data contained within a specific attribute. Thus, the data, and inter-data relationships, can be defined on an attribute-by-attribute basis. The item-id, or attribute 0, of the attribute definition item is a mnemonic "name" for that attribute. This feature is used extensively by the PICK Operating System's English-like data base ACCESS Processor as a user-oriented means to identify the data. For instance, a user could refer to the "LIST-PRICE" rather than "ATTRIBUTE 14." The following is a summary of the attributes in an attribute definition item.

3.5.1 ATTRIBUTE 0, THE ITEM-ID

The item-id becomes the mnemonic label for the attribute being defined.

3.5.2 ATTRIBUTE 1

Within a dictionary file, an item that has the character A in attribute 1 indicates that the following attributes define the nature of the data in the file or files the dictionary points to.

3.5.3 ATTRIBUTE 2

Attribute 2 is the attribute mark count, or AMC attribute. It refers to the number of the attribute in the data file to which the attribute definition item refers. Remember, the item-id gives the mnemonic label the user will refer to the attribute by; this is the actual numerical pointer the system will use to identify the attribute. Since the item-id is attribute 0 for every item, attributes 1-n are logically referred to by the AMC of 1-n. If the AMC is 0, then the item-id is being defined.

A "pseudo" AMC higher than the actual number of attributes in the file being referenced can be used to manage data that is computed, but not really stored in the file. For instance, the attribute "DEALER-PRICE" might not be stored in the file, but would be computed by taking the real attribute "AVERAGE-COST" and multiplying it by the real attribute "DEALER-MARKUP." Once defined, this "virtual data," which is never really stored, becomes as "real" and useful for the user as any that is, in fact, stored as an attribute.

3.5.4 ATTRIBUTE 3

Attribute 3 contains an optional label that will be used as the heading for printout. To save keystrokes, the item-id of the attribute definition item, which is normally the label, might be "PN." Attribute 3 could contain "INTERNAL INVENTORY PART NUMBER CODE" to make the printout more readable for the occasional user. The heading can be defined as having more than one line.

3.5.5 ATTRIBUTE 4

Attribute 4 contains the associative structure code. This code can be used to identify "controlling" and "dependent" attributes. This relationship is used primarily in printout formatting. For instance, if the controlling attribute is suppressed during printout, the dependent attributes will be suppressed too. This relationship extends to attributes containing multiple values. If the 28th value of a controlling attribute is suppressed, the 28th values of each dependent attribute will be suppressed also. Subvalues contained within values are controlled according to the value in which they are contained. A controlling attribute can control many dependents, but a dependent attribute can only have one controller.

When defining a controlling attribute and the one or more dependent attributes it controls, the format is "C" followed by the attribute mark count or "amc" of the dependent attributes (e.g., C;amc;amc; etc.).

When defining dependent attributes, the code D;amc is used in the attribute definition item of each dependent attribute to identify the one attribute that it is controlled by.

3.5.6 ATTRIBUTE 5

Attribute 5 is not used in attribute definition items.

3.5.7 ATTRIBUTE 6

Attribute 6 is not used in attribute definition items.

3.5.8 ATTRIBUTE 7

Attribute 7 contains the conversion specification that is used to convert from processing format to output format. This code causes the data to be processed in a variety of ways before being output. The operation is similar to the process described in attribute 8 and the conversion codes are the same.

3.5.9 ATTRIBUTE 8

Within the PICK Operating System, data can be stored on disk in a variety of ways. For instance, the date is stored as a four-byte code, the time is stored as the number of seconds since midnight, etc. Also, data could be stored one way, but might need to be converted in a variety of different ways for output. A check register might store check amounts in decimal form, but a printout would include a leading dollar sign. The actual check might require a number of leading asterisks. A summary financial report might use that data base, but round up or down to even dollar amounts. Both attributes 7 and 8 provide correlative conversions which are used to convert from the stored format to processing format.

Using these functions, data can be altered from the stored format to an intermediate format for computation, and then to another format for output. Since these functions operate on an attribute-by-attibute basis, different data within one file can be manipulated with complete flexibility.

The following is a summary of standard conversions provided with the operating system:

1. The A Conversion/Correlation - A or arithmetic is used to compute mathematical expressions. A is followed by an expression. A variety of standard arithmetic and relational expressions are available in addition to special functions.

2. The C Conversion - C or concatenate is used to concatenate attribute values. An optional argument can be used to put one or more spaces or other characters between the two concatenated strings.

3. The D Conversion - D or date is used to convert back and forth from the internal format to a number of standard external formats. The date is stored in the system as four bytes representing the number of days from a fixed point in time.

4. The F Conversion/Correlation - F or function is used to compute a mathematical function on attribute values. An A conversion is actually converted to an F at run-time.

5. The G Conversion - G or group is used to extract one or more fields separated by a user-defined delimiter other than one of the normal system delimiters. For instance, G with the delimiter argument of "-" could be used to extract the department number only from an employee file where an employee number was the form department-employee.

6. The L Conversion - L or length is used to extract only data that has a length less than, or equal to, the argument. Multiple arguments can be separated by the semicolon.

7. The R Conversion/Correlation - R or range is used to extract only data that has a length which falls within the range of the argument. The argument is two decimal numbers separated by a comma indicating the range of data that is acceptable. Multiple ranges can be specified by separation with the semicolon.

8. The MC Conversion - MC or mask character is used to convert strings to upper or lower case or to extract alphabetic or numeric characters from a string.

9. The ML and MR Conversions - ML and MR are the mask decimal left or right conversions used to format decimal numbers. Optional arguments give the number of digits to be printed to the right of the decimal point, control the rounding off process, insert commas, cause negative numbers to be printed with "CR" and positive numbers to be printed with "DB," append dollar signs, insert text strings, etc. L and R control whether the result will be printed left or right justified.

10. The MT Conversion - MT or mask time is used to convert the time of day from the internal format to the various standard formats.

11. The MX Conversion - MX or mask hexadecimal is used to convert ASCII character strings to their hexadecimal (base sixteen) representations.

12. The P Conversion - P or pattern match conversion is used to return only data which matches a specified pattern. The argument can call for a specific number of numeric characters, a specific number of alphabetical characters, or a specific number of either. The argument can also call for a specific pattern as in the following example:

    P(3N-2N-4N);(9N)

    This requests a pattern match of three digits, two digits, four digits separated by hyphens (e.g., a social security number). This example also illustrates the use of the semicolon in separating multiple arguments. In this case, the second argument of nine digits without hyphens is also acceptable, so both of the following would satisfy as a match:

    410-96-5644 and 410965664

13. The S Conversion - S or substitute is used to test data to see if it is null or zero. The argument is another attribute number and a string in quotes. If the data is zero or null, then the string in quotes will replace it on output; if it is non zero, then the other attribute will replace it. For instance, an inventory report could print "OUT OF STOCK" whenever the quantity value was zero.

14. The T Conversion - T or text extraction is used to extract a specified number of characters from an attribute. The argument consists of the starting character position and the number of contiguous characters to extract.

15. The Translate Conversion/Correlation - Translate is used to convert data by examining another file. The format of this attribute is T followed by the name of another file. On both input and output operations, data can be converted. The incoming data to be translated represents an item-id in the "other" file. A lookup is performed and the contents of the item-id become the translation. Different attributes of the item-id can be used on the input and output pass. As an example, standard abbreviations can be processed in the following manner: On the input pass, Mastercharge could be abbreviated MC and on the output, MC could become Mastercharge. A savings of 10 characters per entry in the data base is realized without compromising the readability of the printout, and when the terminology "Mastercharge" must be altered to "Master Card," the conversion is simply and quickly realized.

3.5.10 ATTRIBUTE 9

Attribute 9 contains the output specifications for the line printer or terminal screen (flush left or right) and is also taken into consideration during sorting.

3.5.11 ATTRIBUTE 10

Attribute 10 contains the maximum length of values for the attribute. This maximum is for columnar printout purposes and does not represent a limitation on the length of the stored data. Data that exceeds this length will be folded at word breaks during printout.

3.5.12 ATTRIBUTES 11-14

Reserved for system use.

3.6 SUMMARY

The dictionary system is an hierarchical structure which controls many aspects of the PICK Operating System environment.

The system dictionary contains the system programmer's account SYSPROG, the system error message file (ERRMSG), a library of standard PROCs (PROCLIB) (refer to Section 8, PROC Language Processor, for more information), the accounting history file (refer to Section 5, System Software and Utilities, for more information), and the new account file (NEWAC), which is the prototype of an individual master dictionary. The system dictionary also contains D-ITEMS pointing to the individual master dictionaries of each user.

The existence of a master dictionary, defined in the system dictionary, is synonymous with the existence of a user account. The master dictionary contains the user's command vocabulary, on a user-by-user basis, thus making it possible for different users to communicate in different languages, on the same computer, at the same time. The master dictionary also points to file dictionaries.

The existence of a file dictionary, defined in a user's master dictionary, is synonymous with the existence of a file. File dictionaries define the nature and structure of individual attributes within a file, on an attribute-by-attribute basis. These attribute definition items, used in conjunction with the English-like ACCESS Processor, assist in the retrieval process. Attributes can be individually retrieval and update protected (refer to Section 6, System Security, for further information), can give printout instructions, and can define inter-attribute relationships including the existence of computed "virtual" data that is, in fact, not really stored in the file at all.

data base architecture and the access process 4

4.1 INTRODUCTION

The mechanics of the PICK Operating System file structure has been discussed in Section 2, File Structure, and Section 3, Dictionary System. The flexibility of this file structure lays the foundation for the higher level data base management facility within PICK Operating System and English-like ACCESS Processor.

The first part of this section will discuss some of the precepts upon which data base systems operate and the second part will discuss how these are specifically implemented in the PICK Operating System.

4.2 DATA BASE APPROACH

A "data base" is an accessible collection of separate information values. The purpose of a data base system is to manage all of the operational information pertaining to an organization in one coherent way, using one set of standards. Once this has been done, the information can be made accessible, thereby useful, in a controlled manner.

In practice, the data base system replaces separate files and file management code in individual applications programs with the "data base" and related logic. Individual applications no longer access their own files, but simply request data transactions to and from the data base system.

Replacing the file management responsibility in individual programs with one centralized and standardized process provides an organizing force over the total range of data processing activity. Two or more application programs no longer maintain redundant data; they both share the same data base. This process alone has the effect of increasing the accuracy of the data within the system, since updating a value stored in the data base has the effect of updating that value for all users of the data. Integrity constraints on additions and modifications to the data can be implemented in a controlled way. This, too, has the effect of increasing the accuracy of the data base and, by implication, improving the overall performance of the entire data processing system.

The real purpose of the data base approach is to enhance the information content of the data that drives the system. "Information" not only includes a retrieval of stored data items, but analysis based on relationships between them. A data base system is not a file cabinet, but an information resource. True data base systems provide a query facility that allows a user to express one or more relations between data items and request meaningful analysis and output based on those relations. A relational approach to the data base can produce new information where there was none before.

These points are keyed to the usefulness of data base systems in a business computing environment. Running a complex business is, at best, like dealing with a moving target. If a data base system is to assist the decision making process, it must be flexible enough to deal with unanticipated requirements. Such a system can be invaluable to the business executive dealing with dynamic and highly competitive markets.

4.3 ELEMENTS IN A DATA BASE SYSTEM

A data base system can be broken down into roughly three components. The first is the internal data model or storage system. This is how real data is actually stored and physically managed, and this also includes the storage and management of any associated directories necessary for data access. The second is the external data representation or how the user "sees" data and manipulates it. The third is the query language provided for inquiry into the data base. This is one of the most important parts of the system, since the query language is what makes the data base accessible and useful.

4.4 DATA BASE FILE STRUCTURE, THE INTERNAL MODEL

Various data base systems can be characterized by the manner in which files are stored and retrieved. These techniques constitute the internal data model, and greatly affect the other elements of the system, especially formulation of the retrieval language. The structure of the internal data model also has a great impact on how easily the system can be understood and managed. Data base systems are complex, and elaborate internal systems can make a theoretically useful system unmanagable in the real world.

4.4.1 HIERARCHIES AND NETWORKS

Hierarchical systems, as the name implies, have a hierarchy of data. Thus, a vendor number might point to a city, which would point to a part number, which would point to a weight. Such systems are generally represented by tree structures and involve a number of different file types. A simple example is shown in Figure 4-1.

Network systems are similar to hierarchical systems except the linking structure is not a vertical hierarchy or "tree," but a flexible series of links. Unlike hierarchies, networks can have many links connecting data elements. An example is shown in Figure 4-2.

Networks, like hierarchies, normally maintain at least two types of files: "real data" files and "link" files.

Both networks and hierarchies suffer from a number of problems. Their inherent structure is complex, and thus the associated supporting programming must be complex; and they do not model the "real" world very well. Hierarchies are excellent for modeling true hierarchical relationships, but they are only a small fraction of the types of relationships a flexible data base system may be called upon to manage. Network systems are really just a complex way of making hierarchies more flexible and they do model the real world better than hierarchies, yet the complexity of both of these data models makes the formulation of a clear, user-oriented query language all the more difficult, since both systems represent data relationships in several ways using several different file types.

The real Achilles heel of these systems is the fact that relations between data items are intrinsic to the structure and must normally be predetermined in the systems analysis phase and built into the data model before they can be supported in the external level. These systems offer high performance in certain limited applications, but do not satisfy the requirements for a generalized "information resource."

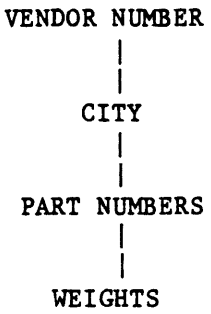


Figure 4-1. Hierarchical Data Representation

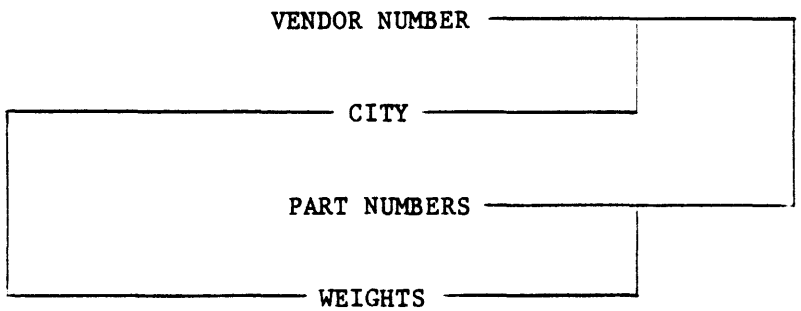


Figure 4-2. Network Data Representation

4.4.2 RELATIONAL SYSTEMS

The PICK Operating System is a relational data base system. The relational approach is based on the principal that "relations" between data elements are the fundamental subject of concern when making a query into a data base, so it is the most useful data model for most types of applications. A relational data model is conceptually like a common "table" with various entries. An example is shown in Figure 4-3.

Using the relational system, many different types of inquiries can be handled easily by examining "rows" and "columns." On the PICK Operating System, the file structure used to store data is also flexible enough to store the dictionary files associated with the data files. In fact, every file on the PICK Operating System is stored using the same file structure as is used when storing data, and this feature provides a welcome level of simplicity.

The PICK Operating System internal file model is discussed in detail in Section 2, File Structure, but the important point here is the close relationship between the internal file structure and the external appearance of the data. This close relationship provides a further level of simplification, and makes the "conceptual leap" between stored data and its meaning much less difficult for the average user to comprehend.

VENDOR NUMBER	CITY	PART NUMBERS	WEIGHTS
xxxx-xxxx	xx	xx-xxxx-xx	xx
xxxx-xxxx	xx	xx-xxxx-xx	xx
xxxx-xxxx	xx	xx-xxxx-xx	xx
xxxx-xxxx	xx	xx-xxxx-xx	xx
xxxx-xxxx	xx	xx-xxxx-xx	xx

Figure 4-3. Relational Data Representation

A PICK Operating System file contains items consisting of an item-id followed by one or more attributes delimited by the attribute mark (character ^). Each attribute can contain one or more values delimited by the value mark (character]). Values can contain one or more subvalues delimited by the subvalue mark (character \). Using this technique, we can have a data model consisting of a flexible number of "rows" and "columns."

The following is an example of a PICK Operating System file shown using the internal file model:

```
SUB-ASSEMBLY-A^5]6]7]8^100]200]214]1065^15]22]5]28
|              |       |                |
ITEM-ID        PART# PRICE              WEIGHT
```

The first attribute delimited by the mark is the item-id. This attribute indicates the following data pertains to "Sub-Assembly-A." The next attribute is the "Part #" and it contains four values or part numbers. The remaining fields also contain four corresponding values. In the PICK Operating System, the structure of this data, and even the mnemonic names of the fields, are stored in the file dictionary. Thus, the fact that the price field is decimal dollars can be defined to the system by various attribute definition items. It is easy to see how the internal model can be transformed into the understandable external representation shown in Figure 4-4.

(item name) SUB-ASSEMBLY-A

(the three attributes)	PART#	PRICE	WEIGHT
(various values)	5	$ 1.00	1.5
	6	$ 2.00	2.2
	7	$ 2.14	.5
	8	$10.65	2.8

Figure 4-4

4.5 QUERY LANGUAGE "ACCESS"

Data base query languages can be divided into two major groups, procedural and non-procedural. A procedural query language is one in which the procedure or method used to come to a conclusion is part of the information the user is required to supply before the language processor can begin to solve the problem. Conversely, non-procedural languages allow the user to express what the results are to be, not how the processor is to go about determining those results. It is generally accepted that the non-procedural approach is more desirable for use in formulating query languages for data base systems since end users find them easier to work with. A procedural language is very much like a "normal" programming language; knowledge of how a file is constructed is almost mandatory before a procedure to read it can be developed. A powerful non-procedural language, like PICK Operating System's English-like ACCESS Processor can be used by a novice user, since no real understanding of the file structure is required. A query concerning the file "Sub-Assembly-A" might be:

```
LIST PARTS-FILE "SUB-ASSEMBLY-A" PART# PRICE > "$2.00"
```

The results might be displayed:

```
SUB-ASSEMBLY-A

Part#...PRICE...
7          $ 2.14
8          $10.65
```

The fact the file also contains information concerning the weight of every part is irrelevant to the query. To persons in the shipping department, the weight, and possibly the warehouse location or vendor number, might be of great concern. The price field might be password protected so certain personnel could not access price data. A procedural query language would greatly inhibit the ability of users to inquire into the data base freely, without concerning themselves with extraneous or security-sensitive information.

4.6 ACCESS QUERY LANGUAGE

The PICK Operating System's English-like ACCESS Processor is a special purpose data base inquiry language. The ACCESS Processor provides a programming tool which can be used to quickly inquire into, and generate reports from, the data base. The processor can be used by non-programmers to formulate simple, ad hoc inquiries at a CRT terminal, or by programmers or trained personnel to generate extremely complex reports.

The syntax of the PICK Operating System ACCESS Processor is close to standard English in that commands called "verbs" are action-oriented operations and generally "do what they say." The file system provides great flexibility in providing mnemonic "names" for various data items so that the user has to deal with a minimum of abstract concepts.

Stored ACCESS Processor commands can be used to efficiently generate all the normal working reports required by a given application. When the format of these reports requires revision, the task can be accomplished in a minimum amount of time. This is in sharp contrast to conventional programming languages, which might require several programmer hours to effect even a minor format change.

When a special, one-time situation arises, the ACCESS Processor can produce a custom report to assist the management decision making process. For instance, a company has the opportunity to replace a sub-assembly in a product they manufacture with a newer integrated circuit which can be purchased at a lower cost. Switching over to this integrated circuit immediately can give them the price advantage in a bid for 100,000 systems to a large customer. However, the present inventory of parts used in this sub-assembly must then be scrapped. Complicating matters further, some of the parts can be used in other products, so those parts would not have to be scrapped, just amortized over a longer period of time.

The management team charged with making a decision on this matter wants to know the following by five o'clock:

> What parts used in sub-assembly SA-19523 are in stock and not used in any of our other products, and what is their wholesale value?

The management served by a conventional inventory system would probably be told that an answer by five o'clock was simply impossible. A manager using the PICK Operating System who was familiar with the ACCESS language processor might formulate the following inquiry himself. If not, a properly trained assistant could.

```
LIST PARTS-FILE "SA-19523" WITH SUB-PARTS-QTY > "0" AND WITH NO
 SUB-PARTS SUB PART# WHOLESALE-COST QUANTITY TOTAL INV-VALUE
```

The report might look something like this:

```
PARTS-FILE.....SUB-PART#...WHOLESALE-COST..QUANTITY..INV-VALUE..
SA-19523       SB-1350              4.532       563     2551.52
               SB-1468             10.250        18      184.50
               SB-3971               .511      1587      810.96
***
                                                       3546.98
```

The management team has the cost data they were looking for in order to make an intelligent decision. And the accounting department has satisfied a complex requirement without spending an inordinate number of programmer hours developing special software for a one-shot situation.

4.7 PICK OPERATING SYSTEM ACCESS LANGUAGE STRUCTURE

The theoretical foundation of the ACCESS Processor is relational calculus, a mathematics of relations. The language is inherently non-procedural; the desired results are simply stated, as in:

```
LIST PARTS-FILE "SUB-ASSEMBLY-A" PART# PRICE > "$2.00"
```

which simply asks for those parts with a price greater than $2.00. The statement does not say how the results are to be determined.

Most of the mathematical roots have been hidden in the implementation of the ACCESS Processor to make it as user oriented as possible. For example, the concept "greater than" can be expressed with the usual > sign, or the letters "GT," but for time concepts where a date in the future is technically "greater than" and a date in the past is "less than" today's date, the words before and after can also be used. Thus, an easily understood query can be formed:

```
LIST ACCOUNTS-RECEIVABLE WITH ITEM-DATE BEFORE "JUNE 30 1980"
```

The language also has a number of extended functions that are not, strictly speaking, data base query functions. These functions include sorting, counting, statistical derivatives, and complex printout formatting.

The ACCESS Processor is also extended by way of interface to the rest of the PICK Operating System. Most of the ACCESS functions can be accessed by other processes, such as BASIC (Section 9) and PROC (Section 8). This ability to interface expands the ACCESS Processor from a utilitarian report generator to a powerful system tool that can be used in conjunction with other system functions and thereby expand the file management and data base handling ability of any other processor.

4.8 USING THE ACCESS PROCESSOR

The PICK Operating System's English-like ACCESS Processor accepts commands from the terminal control level or from a PROC command (Sections 7 and 8). These commands consist of verbs which are identified as items in an individual user's master dictionary, and modifiers and options standard to those commands. The options constitute reserved words, but the verbs can be renamed on a user-by-user basis to reflect appropriate jargon, or otherwise enhance the clarity of a special application.

A query command acts on a file via the file dictionary. Many of the attribute definitions set up in a file dictionary are for the use of the ACCESS Processor. The item-id of an attribute definition provides a mnemonic label for a specific attribute in the data file. Thus, when formulating a query, the user can refer to data fields by names like "part-number," "quantity," and "price," rather than some abstract coding structure. A query statement consists of a valid command-verb followed by a filename followed by an optional list of selection criteria, sorting keys, output specifications, etc.

Selection criteria limit a command's action to those items which meet the criteria. A variety of relational operators are available. Sorting keys can be the item-id, or any attribute. Multi-keyed ascending and descending sort capability is built into the ACCESS Processor. Output specifications indicate which attribute definition items in the dictionary are to be consulted when formatting the material for output. Also, various attributes can be suppressed. Print limiters suppress the output of values and sub-values not meeting certain selection criteria. Modifiers and options control the format of the printout, double/single spacing, optional columnar headers, intermediate totaling, page formatting, etc.

4.9 ACCESS LANGUAGE COMMAND VOCABULARY

The verbs that comprise the standard vocabulary of the ACCESS Processor are briefly outlined below. Each verb name is an item-id in the users' master dictionary so the command vocabulary can be modified to suit special applications on a user-by-user basis.

4.9.1 LIST, SORT, LIST-LABEL, AND SORT-LABEL VERBS

These verbs produce formatted output. LIST reads items from the file in sequential order; SORT produces a sort by item-id or one or more attributes. Both verbs examine the maximum lengths of attributes which are to be output by consulting attribute 10 of the attribute definition items in the file dictionary. Then, taking into consideration the maximum width of the terminal screen or line printer page, a listing is produced in either columnar format with attribute names across the top of each page, or non-columnar format with attribute names printed down the left-hand side next to each field of output data.

The LIST-LABEL and SORT-LABEL verbs allow data from more than one item to appear across one line. This is used in producing four-across mailing labels. The system prompts for the count across, number of rows per label, number of blank lines between labels, etc. The action of the LIST-LABEL and SORT-LABEL verbs in selection and sorting is otherwise identical to LIST and SORT.

When sorting, the BY-EXP, or "by-explosion," modifier can be used to "explode" an attribute into multiple values (delimited by the value mark]) and cause sorting on those values.

4.9.2 COUNT VERB

The COUNT verb counts the number of items which satisfy any selection criteria. If no selection options are used, the whole file will be counted and the total number of items in a file will be the result of the count.

4.9.3 SUM AND STAT VERBS

These verbs are more sophisticated versions of count. SUM will give the sum of all specified attributes which satisfy the selection criteria. STAT will also give the count and average as well as the sum.

4.9.4 THE SELECT AND SSELECT VERBS

SELECT and SSELECT both generate lists of data elements. These lists, which are subsets of the total file, represent items which met the selection criteria. The next query command input will act on this list rather than the entire file. The list may also be saved and used later via the SAVE-LIST, GET-LIST, and DELETE-LIST commands. SELECT generates a normal list; SSELECT generates a list sorted on whatever keys were specified.

4.9.5 OTHER VERBS

Other query verbs are T-DUMP and T-LOAD, magnetic tape handling verbs; ISTAT and HASH-TEST, disk space utilization analyzers; and LIST-ITEM and SORT-ITEM, which dump items to a user's terminal or a lineprinter according to selection criteria, sorting criteria, and print limiting criteria.

4.10 ACCESS LANGUAGE COMMAND MODIFIERS AND OPTIONS

All of the ACCESS Processor verbs are normally used with modifiers and options. Such modifiers typically include instructions on selection of a subset of the file, sorting on specific keys, and instructions on how an individual printout is to be formatted. The following is a brief discussion of the most important modifiers and options.

4.10.1 RELATIONAL OPERATORS

Relational operators are available to construct selection criteria when inquiring into a file. Such operators are:

Operator	Expressed as:
Equality	= or EQ
Greater than	> or GT - for math concepts AFTER - for time concepts
Less than	< or LT - for math concepts BEFORE - for time concepts
Less than or equal to	<= or LE
Greater than or equal to	>= or GE
Not equal to	# or NE or NOT or NO

The logical connectives AND and OR bind two relational operators together. If a query consists of two relational operators connected by AND, then both must be true for the statement to be true; if they are connected by OR, then either may be true and both may be true for the statement to be true. Using relational operators and logical connectives, extremely complex selection criteria can be defined. For instance, the following statement requests all items which are equal to "NEW" or "NEW0" or are greater than "NEW2" and less than "NEW5" in the file called "NEW-ONES."

```
LIST NEW-ONES = "NEW" OR "NEW0" OR > "NEW2" AND < "NEW5"
```

4.10.2 WITH AND IF MODIFIERS

WITH and IF are synonymous modifiers that refer to an attribute of an item. They may be followed by any of the relational modifiers. If an attribute consists of multiple values, each value is tested. The modifiers EACH, EVERY, and NO can also be used. EACH and EVERY are synonyms. In the following example, the attribute called "DATE" is tested in the file called "NEW-ONES." There may be more than one date stored in each item, but no date may fall outside the range of 01/01/79-01/01/80. Thus, any item with a date outside this range is excluded, regardless of how many values within that item satisfy the selection criteria. Those items with dates between 01/01/79 and 01/01/80 are acceptable except for 03/26/79.

```
LIST NEW-ONES WITH EVERY DATE AFTER "01/01/79" AND BEFORE "01/01/80"
AND WITH NO DATE = "03/26/79"
```

4.10.3 STRING SEARCHING

Any item-id or attribute value may be specified as part of the selection criteria. This selection can be accomplished by relational operations like less than or greater than, or by searching and matching a specific string. String searching can be invoked by the use of the character [and ^ in any inquiry. The [character specifies "any number of any character" and the ^ specifies "any one character." Thus, [ING would find a match with any string ending in ING and ^^^ING would find a match with any string starting with any three characters and ending in ING.

4.11 SELECTION OF OUTPUT

Selection criteria are not automatically output, and the user may want to see other attributes whether or not they were involved in a selection or sorting process. Additional attributes may be referred to by simply typing in the appropriate attribute name as defined in the file dictionary. Thus, the following query acts on items with the attribute "QUANTITY" of less than 10, but requests that only the part number and vendor telephone number be printed out.

```
LIST PARTS WITH QUANTITY < "10" PART-NUMBER TELEPHONE
```

4.12 FORMATTING FOR OUTPUT

The ultimate goal of any ACCESS Processor command is to output reports of some type, either on the lineprinter or the screen of the user's terminal. Some commands have built-in formatting, like the columnar/non-columnar output of the list verb; and these defaults aid in the display of quick inquiries. However, the query language provides a full range of print output formatting tools that can generate even the most complex reports from the data base. These formatting tools work in close conjunction with the attribute definition items in the file dictionary, as do all of the ACCESS Processor commands. The following summarizes the major formatting options.

4.12.1 HEADINGS AND FOOTINGS

When using the LIST, SORT, LIST-ITEM, and SORT-ITEM commands, headings and footings can be specified to replace the default heading (page number, time, and date). After the normal query commands and selection criteria are entered, the command HEADING or FOOTING followed by a string in double quotes will take that string as the heading or footing for each page of the report. Single quotes can surround special commands within the heading to generate linefeeds, center lines, insert the date or the time and date, page numbers, the file name the report is taken from, and the name of the attribute presently flagged by the breaking process described below.

4.12.2 BREAKING

The breaking process is used to insert a blank line, a special printed comment, or a sub-total into a report periodically based on some value or value change.

The BREAK-ON modifier takes as an argument an attribute name. Every time the value of this atribute changes, a "break-on heading" line with asterisks in the column of that attribute will be generated or an optional text string can be inserted instead. Also, when a break occurs, the name of the attribute, or the value of the attribute at the breakpoint, can be inserted in a heading or footing, thus providing a quick reference type of indexing at the top or bottom of the page. Other options include the ability to force a new page to begin on the break.

4.12.3 BREAK-ON TOTALING AND GRAND TOTALING

The TOTAL modifier will print the total of all the attributes since the last break. Another attribute can be totaled instead of the attribute being monitored for break-on purposes. For example, a report could break on a certain inventory class, but total the wholesale price attribute. Options include the ability to surround the total with user-defined text. The GRAND-TOTAL modifier inserts the grand total on the last page of the report and generates a page-eject.

4.13 SUMMARY

The PICK Operating System data base file structure is founded on relational data base principles, resulting in a high level of storage efficiency consistent with complete data management flexibility. The system incorporates a high-level, non-procedural ACCESS Processor based on relational calculus and includes extended functions like sorting and counting, and a user-oriented English-like syntax. It can be easily used by non-programmers to formulate simple inquiries into a file, yet provides a full array of powerful features the experienced programmer can use to assemble, debug, and maintain complex applications software. Using this query facility, data base reporting requirements can be met within reasonable time periods, without the costly custom software development normally associated with the use of traditional programming languages.

system software and utilities 5

5.1 INTRODUCTION

In addition to the operating system task scheduler, memory management, and data base management software, the PICK Operating System provides a number of additional "utilities" which enhance the overall operation of the system. Many of these functions are normally found only on elaborate mainframe computers. Although PICK Operating System is priced like a mini-/micro-computer, it is commonly used in timesharing services supporting 16 to 32 telecommunications ports per CPU. The security and system accounting features go a long way towards making these difficult applications possible.

5.2 SYSTEM USAGE ACCOUNTING

One of the standard files that constitute the operating software is the Accounting History file. This file is used by the operating system to accumulate statistics on each individual's use of system resources. This file is divided into two sections: one part for "active user items," defining users who are presently active on the system; and one part for "accounting history items," defining past history.

Active user items include the name of the user, the port logged onto, and the logon time. This data can also be used to send messages to a specific person by "finding" the port to which she/he is logged.

Accounting History consists of items that include the account name of the user as defined in the system dictionary, the channel or port number to which the user was logged on for that session, the date and time logged on, the total connect time, CPU time charge units in tenths of a CPU second, and the number of pages routed to the lineprinter.

Since the Accounting History file is structured like any other file on the system, the ACCESS Processor can be used to generate reports on system loading by port number, average connect time per user, average number of sessions per account, etc., as well as totals for customer billing or internal chargebacks.

5.3 INDIVIDUAL USER ACCOUNTS

Each user of the PICK Operating System can be assigned an "account." This process consists of defining the user as an item in the system dictionary, which will point to a master dictionary for that user. Initially, a prototype master dictionary is copied to form the user's master dictionary. The account information for each user in the system dictionary consists of the logon name, a logon password, and the file access codes for read and write privileges (refer to Section 6, System Security).

5.4 MULTIPLE USER ACCOUNTS

File synonym definition items can be established in the system dictionary to allow multiple users to have access to the same "account." In this case, the concept of an "account" is a group of files and the "user" is an individual with access to those files. This distinction can be used to allow multiple users, like a group of people in the accounting department, to have controlled access to one set of files.

The file synonym definition item for each of these users points to the same master dictionary, but each user will have a separate password, system privileges level, and update and retrieval codes. Thus, some users might be able to access all of the files in the account and some might be restricted to certain files or read-only privileges. Figure 5-1 shows an item in a system dictionary that identifies an account called "accounting." Figures 5-2 and 5-3 show two synonym definitions that give users "JWB" and "RHA2" access to "accounting" with separate passwords and privilege levels.

```
ACCOUNTING  <-------------- The Accounting Files
001  D  <-------------------------------------------------This is the file
002  05412  <-------------  Base of Master Dictionary     definition item that
003  3  <-----------------  Modulo                        establishes a master
004  1  <-----------------  Separation                    dictionary to the
005  ARAPPRGL  <----------  Retrieval Codes               accounting files
006  ARAPPRGL  <----------  Update Codes
007  LG  <------------------------------------------------ Password*
008  SYS2  <--------------  System Privileges Level
009  RL  <----------------  User Type
```

*Passwords are in fact hash-coded to maintain security and are displayed here only for clarity.

Figure 5-1

```
JWB  <--------------------  The Account of user "JWB"
001  Q  <-------------------------------------------------This is the synonym
002  ACCOUNTING  <---------  "Host" Account                 definition item that
003  (Not Used)                                            establishes user JWB
004  (Not Used)                                            as someone who can
005  ARAPPR  <-------------  Retrieval Codes                access the accounting
006  ARAPPR  <-------------  Update Codes                   files
007  JWB  <----------------  Password*
008  SYS1  <---------------  System Privileges Level
009  R  <------------------  User Type
```

*Passwords are in fact hash-coded to maintain security and are displayed here only for clarity.

Figure 5-2

```
RHA2  <--------------------  The Account of user "RHA2"
001  Q  <-------------------------------------------------This is the synonym
002  ACCOUNTING  <---------  "Host" Account                 definition item that
003  (Not Used)                                            establishes user RHA2
004  (Not Used)                                            as someone who can
005  AR  <-----------------  Retrieval Codes                access the accounting
006  <---------------------  Update Codes (None)            files
007  RA  <-----------------  Password*
008  SYS1  <---------------  System Privileges Level
009  RU  <-----------------  User Type
```

*Passwords are in fact hash-coded to maintain security and are displayed here only for clarity.

Figure 5-3

In Figure 5-2, user JWB has been defined as a user who can access the files in "accounting." The original file definition item in the system dictionary that sets up the "accounting" account has a set of retrieval codes and update codes that control access to the various files in accounting (refer to Section 6, System Security, for further information). User JWB has only one of these codes in the retrieval and update attributes, so access to only files locked with "ARAPGL" is possible for user JWB. JWB has a private password and a lower system privileges level than the original account.

In practice, the main account definition (a "D" in attribute 1) will be set up for the person responsible for all the activity in that account. This definition will include all the retrieval and update "keys" for all the files in that account and will have a high system privileges code. The user population that requires access to this account will be defined by synonyms (an "A" in attribute 1) and will have low system privilege levels and a subset of the update and retrieval "keys" for selective file access. Each user can have a private password.

In Figure 5-3, user "RHA2" is defined in a similar manner to user "JWB." However, RHA2 can access only those files locked with "AR" and cannot upate any files at all.

5.5 LOGGING ON

Simply typing a valid account name initiates the logon process. Upon logon, the user's logon name is placed in the active user section of the accounting file. If the entry in the system dictionary that describes the user's logon name contains a "U" in attribute 9, use of the system will be reported in the accounting history file; if attribute 9 contains an "L," no accounting of system use will be kept. This would normally be the case for the system operator and other "free" users. An "R" will force the logon PROC to be re-started should the user's program abort at any time during the session. This feature can be used to keep a security-restricted user in one account doing one task. "RU" and "RL" combinations are also valid to indicate charge and non-charge status to "R" class users.

5.6 THE CHARGES AND CHARGE-TO COMMANDS

The CHARGES command simply displays the accumulated charges for any session. The CHARGE-TO command allows a user to separate charges into different projects by typing CHARGE-TO plus any meaningful label. The user does not have to logoff and logon again; the CHARGE-TO command is executable anytime during a logged in session at TCL level (refer to Section 7, Terminal Control Language, for further details). The different projects are treated as ordinary line items in the Accounting History file, but instead of using the logon name only, the logon name plus an asterisk plus the project name is entered. Thus the account for user XYZ might look like this:

ACC.....DATE.	TIME...	CONN...	UNITS..	PAGES
XYZ 07/01	16:15	00:10	9	1
XYZ*PROJECT-10 07/02	16:27	02:15	89	11
XYZ*BASIC-PROGRAMS 07/05	15:58	01:49	174	6

The CHARGES and CHARGE-TO commands, of course, have no relevance for "L" class users described above.

5.7 SYSTEM BACKUP TO MAGNETIC TAPE

The FILE-SAVE command causes disk files to be copied to magnetic tape. Options include the ability to copy individual files or the entire data base. The FILE-SAVE process will also generate a statistical report on the data base. This file, called the STAT-FILE, contains one item for each file on the system. The report is broken down by account with the following information about each file: the total and average item size, the total and average number of items per group, the utilization of file-space including actual stored data and reserved but unused space. Each account includes the total number of items, total bytes, total frames, and total file errors.

The FILE-SAVE command steps the user through a sequence of questions and answers that make the backup process as simple as possible. The STAT-FILE can be directed to a line printer for hardcopy reference and the magnetic tape reel can be given a unique label in the first record, normally the date of the FILE-SAVE.

The ACCOUNT-SAVE is like the FILE-SAVE process except that it saves only one user account rather than the entire system.

5.8 SYSTEM ERRORS AND MEMORY ERRORS FILE

A file called SYSTEM-ERRORS is reserved for the automatic logging of disk errors. The system operator can examine this file periodically to check on the performance of the system.

5.9 OTHER SYSTEM COMMANDS

Many other commands and miscellaneous functions are available, some for general use and some for system maintenance normally performed only by the system operator. The following is a rundown of some of the more important ones.

5.9.1 LOGON

LOGON is initiated by typing a valid account name at a logged off terminal. The system then asks for a user-id and password before allowing execution of any commands.

5.9.2 OFF

OFF terminates the user's session by deleting the entry in the active user's section of the accounting file and placing it in the Accounting History section. The user is presented with a display on the terminal showing the total connect time, the number of charge units (tenths of a CPU second), and the number of line printer pages generated during the session.

5.9.3 SYSTEM LOGON MESSAGE

A logon message is stored as an item called LOGON in the system error message file (ERRMSG) and is displayed upon logon. This can be used to "broadcast" messages like "Line Printer 3 Down for Cleaning, 2PM to 3PM Today" to everyone using the system. Or, an attractive message like "Welcome to XYZ Corporation Data Processing Department" could be displayed instead.

5.9.4 USER LOGON PROC

If the user's master dictionary contains a PROC of the same name as the user account, that PROC will be automatically invoked upon successful logon. This PROC normally contains attributes which set terminal characteristics like maximum line length, backspace echo routines, etc., but it can also specify another PROC or program to be run immediately after the terminal commands are set.

An "R" in attibute 9 of the user account item in the system dictionary will force the logon PROC to be restarted should the user's program abort at anytime during the session. This is very useful when a restricted employee is assigned to do only one thing, like accounts payable. When the employee logs on, the accounts payable program is automatically invoked without unnecessary, and potentially confusing, steps.

5.9.5 WHO COMMAND

WHO displays the port number the user is logged into. WHO followed by a number displays the user-id of the user at that port number.

5.9.6 LISTU COMMAND

LISTU displays the account name of all users presently logged on to the system, their logon time, and channel number.

5.9.7 MSG COMMAND

MSG allows a user to send a message to another user by user name or port number. Users with privilege levels of 2 (refer to Section 6, System Security, for more information) can "broadcast" a message to all users.

5.9.8 SLEEP COMMAND

SLEEP will put a terminal to sleep for a specified period of time or until a given clock time. SLEEP can be used in a PROC before other commands to delay those other commands until a specified time. For instance, an ACCOUNT-SAVE could be run at 11:30 PM each night by invoking a command at 5:00 PM.

5.9.9 LISTVERBS AND LISTPROCS COMMANDS

These commands examine a specific dictionary and list all verbs (non-PROC commands) or all PROCs in that dictionary.

system security 6

6.1 INTRODUCTION

The PICK Operating System security includes four levels of password and code protection that guard the computer system and data base from unauthorized tampering. Each user is identified to the system by establishment of a user-id in the system dictionary. This user-id is the item-id of an item in the system dictionary and some of the attributes of the item point to the user's master dictionary and some tell the system how accounting for that user is to be handled. A number of attributes are reserved for special codes and these are discussed below.

By establishing synonym definition items in the system dictionary, different levels of security can be assigned to users logging on to the same account.

6.2 PASSWORD FOR LOGON

Before a user can log on to the system, a password is requested. This password is stored in attribute 7 of the system dictionary as part of the user identification and can be as long or as short as desired.

6.3 SYSTEM PRIVILEGES LEVEL

The PICK Operating System includes a three-level system privileges code. One is assigned to each user and is stored in attribute 8 of the system dictionary as part of the user identification. Level 0 cannot update a master dictionary or use magnetic tape. Level 1 cannot use the debugger, the dump processor, the assembler and loader, or run FILE-SAVE and FILE-RESTORE. Level 2, the highest level, is unrestricted. However, some commands can still only be executed from the system programmer's account, SYSPROG.

6.4 FILE UPDATE AND RETRIEVAL PROTECTION CODES

Attribute 5 in the system dictionary contains a special code called the retrieval code and attribute 6 contains the update code. These are password "keys" and each user can have one in either or both of the retrieval or update attributes. There are corresponding codes in each file dictionary in attributes 5 and 6 of the file definition or "D" item and synonym definition or "Q" item (refer to Section 3, Dictionary System, for more information). When a file is created, these "lock" codes can be entered into the file dictionary and only those persons with matching "key" codes in their user-id item can gain access to that file. Using these codes, some users can be given retrieval access only, some users can have update access only, and of course some users can do both. This protection is monitored by every processor that can access a file. Retrieval processors will only check the retrieval codes for a match; other processors that can both retrieve and update a file, like BASIC, will check the update codes on a write operation.

Codes are examined from left to right for a match, so a hierarchical "mask" can be constructed for very elaborate security schemes. The following gives an example:

File Protection Code	Users Code	Evaluation
AR	ARAPGL	ACCESS OK
ARAP	GL	ACCESS DENIED
GL	ARAPGL	ACCESS DENIED

6.5 SUMMARY

Since access is controllable on the file level as well as the system level and file access is monitored separately for both retrieval and update operations, even the most complex security requirements can be easily satisfied by the PICK Operating System. By individual assignment of passwords on a user-by-user basis and a corresponding file-by-file basis, careful control of data base use can be established in even the most dynamic data processing environment. System security is further enhanced, in a more general way, by the use of a three-level restriction on sensitive commands, the restriction of very sensitive operations to the system programmer's account, and by the use of a password to gain initial entry into the system.

terminal control language 7

7.1 INTRODUCTION

Terminal Control Language or TCL is the "monitor" or "command processor" level of the PICK Operating System. TCL is present at each terminal at system start-up prior to logon, but no function except logon is valid. Once logged on, full TCL is invoked and further interaction is initiated from it. When a process is complete, the user is returned to TCL level. The command vocabulary of both the PROC and ACCESS language processors (refer to Section 8, PROC Language Processor, and Section 4, Data Base Architecture and the ACCESS Process, for more information) is invoked from TCL. Movement from program to program within a PROC, or processing initiated by an ACCESS statement, is under full control of the related interpretive process, but when the process is finished, the user is once again returned to TCL.

7.2 THE TCL COMMAND VOCABULARY

The TCL prompt symbol is >. Once a user is at TCL level, any valid command may be entered. The command vocabulary of TCL is made up of items in the user's master dictionary and basically consists of four types of commands; those which do not access files, those that do access files, ACCESS Processor commands (which also access files), and PROC commands. None of these commands are inherent in TCL. For instance, data base queries are the command of the ACCESS Processor which happen to be executed at TCL level. PROCs are stored procedures interpreted by the PROC interpreter, which for the most part is governing the execution of logic in other system programs.

Items in the user's master dictionary can include synonym items, which allow a user to customize his command vocabulary at the TCL level. This flexible feature can allow "jargon" commands to enhance a special application, or multiple language commands to be run in different accounts on the same system.

7.3 SOME COMMON TCL COMMANDS

ACCESS Processor commands which execute at TCL level are discussed in Section 4, Data Base Architecture and the ACCESS Process; PROCs are discussed in Section 8, PROC Language Processor. Since TCL is more of a classic "command processor" which passes control to other program modules, there are no commands which are truly inherent to it, in fact all of the system commands and processes covered in other sections can be initiated from TCL. Some commands which are not strongly associated with another processor are discussed below.

7.3.1 SETTING TERMINAL CHARACTERISTICS

The TERM command establishes the terminal handling protocol for a particular user. Criteria like line length, number of printed lines per page or screen, number of blank lines, line feed/form feed delay, backspace control procedures, cursor addressing procedures, etc. are set using this command. These criteria are also prestored and executed automatically by the logon PROC, so they do not have to be repeated every session.

7.3.2 SETTING TABS

The TABS command allows input tabs to be set for CRT terminals and input and output tabs to be set for hardcopy terminals.

7.3.3 COPY

COPY will "copy" a file (or item) to a CRT terminal, printer, or another file. This is a quick way to see the contents of a file without making an ACCESS Processor inquiry.

7.3.4 SENDING MESSAGES

MSG allows a user to send a message to another user by user name. Users with privilege levels of 2 (refer to Section 6, System Security, for more information) can "broadcast" messages to all users.

7.3.5 TAPE ATTACH

The T-ATT function "attaches" a magnetic tape unit to a user's terminal in preparation for any following read and write commands. Since the tape drive is an inherently single user device, attach will lock out other users until the T-DET detach command is issued. For more information on the operation of the tape drive, refer to Section 12, Magnetic Tape System.

PROC language processor 8

8.1 INTRODUCTION

The PROC or stored procedure language is an interpreter which allows the storage and execution of a lengthy series of commands or operations. Once established, these operations can be invoked simply by typing the name of the PROC. Some of the PICK Operating System commands, CREATE-ACCOUNT for instance, are PROCs stored in the system PROC library.

8.2 THE STRUCTURE OF A PROC

A PROC is stored as an item in the user's master dictionary and is a file like any other on the PICK Operating System. The item-id is the name, and attribute 1 consists of the letters "PQ" which signal that the following attributes are to be processed by the PROC interpreter. Following attributes are PROC statements, one statement per line. Lines can optionally be numbered to facilitate branching within the PROC program. Once stored, the PROC becomes a one-word command which may be executed by typing the PROC name (item-id) at TCL level (refer to Section 7, Terminal Control Language, for more information about TCL). This one-word command can be followed by arguments which are passed to the PROC for processing.

8.3 PROC COMMAND LANGUAGE CAPABILITIES

PROC provides four variable length buffers, two input and two output. Commands provide for the transfer of data from one buffer to the next and then for passing the contents of one of the output buffers "outside" the PROC to another program or another PROC. Once this program has finished processing, control and/or data is passed back to the PROC. Other commands can test relational conditions such as equality (=), no-equality (#), less than (<), greater than (>), string length, and pattern matching conditions (for example, a number followed by three letters), then conditionally pass control to other lines within the PROC.

Functionally, any command or process available on the PICK Operating System at TCL level is available to PROC. Many of the system programs, such as the line printer spooler, the English-like ACCESS Processor, BASIC, and the text editor, which normally require human interaction, have pre-established protocols for interacting with PROC. For instance, if an error occurs during processing under the control of a PROC, the error code is placed in the input buffer. The PROC can be programmed to test the value of this error and act accordingly.

8.4 SCREEN FORMATTING WITH PROC

One of the most powerful features of the PROC language is screen handling. PROC commands can position the cursor, output prompting messages, manage terminal attributes like background/foreground, blinking, etc., all independent of the terminal driver. Using these tools and the conditional testing commands, elaborate input forms can be built which can verify operator input, thus providing a convenient interface to the data base. Since PROC can invoke the ACCESS Processor commands (refer to Section 4, Data Base Architecture and the ACCESS Process, for further details), data base input inquiry, retrieval, and report generation can all be customized using PROCs.

8.5 SUMMARY

The PROC language can be used to automate repetitive and extremely complex sequences of interaction with the system, and is one of the most important software tools available on the PICK Operating System computer. Although deceptively simple, it is a highly utilitarian language which is easy to use and allows rapid development of customized, user-oriented, system commands.

BASIC language processor 9

9.1 INTRODUCTION

The PICK Operating System includes a BASIC language processor as a general purpose programming tool. The PICK Operating System BASIC is an extended version of standard Dartmouth BASIC, the very popular programming language. Since most computer professionals are at least acquainted with BASIC, and many documents discuss the features of this versatile language, this section will cover only those functions of the PICK Operating System BASIC that are specifically unique, or are otherwise standard functions which strongly interact with other PICK Operating System unique software or hardware.

9.2 RE-ENTRANT CODE

The BASIC processor generates re-entrant codes which can be shared among a number of users. In practice, this means that if a program is used by a number of users simultaneously, only one copy of the program must be in memory.

9.3 SOURCE FILES

Program source listings are items in a file. The typical user will have one file for all programs and each item will contain one program. The item-id is the name of the program and each line of the program is an attribute. This is the standard convention used with the text editor discussed in that section.

The object code produced is written to disk and a special pointer is written to the dictionary of the source file. This pointer contains the location of the object code on disk. Figure 9-1 diagrams this structure.

As Figure 9-1 shows, each object code module is stored independently in a contiguous segment of one or more frames on disk, while the source code is stored as an item in one file. Both are tracked by one dictionary. The compiler creates the pointer for each object module in the file dictionary associated with the source. The item-id is, or course, the name of the program. Attribute 1 of this special file definition item is "CC" for compiled code. Attribute 2 is the frame-id of the base of the compiled code. Attribute 3 is the total length in frames of the compiled code. Attribute 4 is always left null and attribute 5 is the time and date of the compile.

9.4 COMPILER FEATURES

The compiler includes a number of options to assist the programmer in the development and debugging of BASIC programs. All options that produce listings can be optionally directed to the line printer.

9.4.1 LIST OPTION

The LIST option generates a line-by-line list as the program is compiled, including error message.

9.4.2 LIST ERRORS ONLY OPTION

The LIST ERRORS ONLY option prints a list of the error lines in the source, the source line itself, and a description of the error.

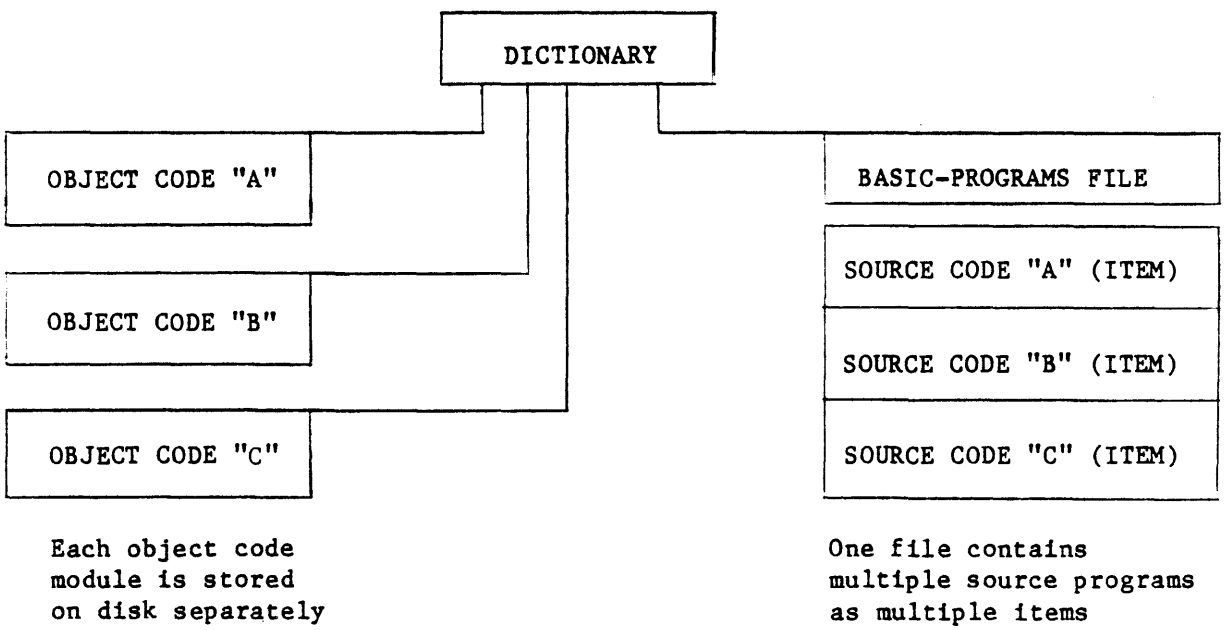


Figure 9-1

9.4.3 ASSEMBLED CODE OPTION

The ASSEMBLED CODE option generates a listing of source statements and the one or more BASIC op codes generated by the compiler for that statement. BASIC op codes are a form of "pseudo assembler" generated by the compiler.

9.4.4 CROSS REFERENCE OPTION

A special file called BSYM must be defined in the user's master dictionary before the BASIC compiler cross reference option can be used. This file is used exclusively to store the cross references of labels and variables when a program is compiled. Each item-id in this file is a variable or label name, and the one or more attributes that follow contain the line numbers in which that label or variable appears. The ACCESS Processor inquiry SORT BSYM BY LINE-NUMBER LINE-NUMBER will generate a printout from this file. This file is cleared before each compilation that specifies this option.

9.4.5 MAP OPTION

The MAP option generates a variable map and a statement map showing where program data is stored in the user's workspace. The variable map lists, in decimal, the offsets from the top of the program of each variable (each variable is "tokenized" into 10 bytes). The very powerful statement map feature lists which statements are located in the operating system buffer frames. The virtual memory manager of the PICK Operating System, like any virtual memory computer, brings only a few program frames into memory at a time. This "paging" makes it possible to run very long programs in a small amount of memory. One of the drawbacks of paging in a heavily loaded multi-user system is a phenomenon called "thrashing," where a program loops back and forth between two sections of code which, because of a high user load, cannot be loaded into memory at the same time. As this looping continues, the virtual memory manager has to "page in" and "page out" the executing code so much that little real work gets done. Analysis of the statement map can be used to carefully group program statements, minimize thrashing, and optimize any given program.

9.5 EXECUTING BASIC PROGRAMS

Once a BASIC program has been compiled, it can be run by using the RUN command followed by the name of the dictionary containing the file definition item pointing to the object code.

The CATALOG command will create a pointer to the object code in the master dictionary of the user. When this is done, the user no longer needs to use the run command; simply typing the name of the program is all that is required. From the user's point of view, the program becomes a system command or "verb" like any other.

9.6 FILE HANDLING IN BASIC

The PICK Operating System BASIC provides a number of unique features for file handling that take advantage of the data base management functions in the PICK Operating System file manager. The BASIC programmer using the PICK Operating System has a set of already developed, application independent data management tools that can significantly reduce development time.

Section 2, File Structure, discusses the operating system's management of disk files. However, a brief review is appropriate at this point.

A PICK Operating System file contains one or more items. These items are identified by an item-id. Items contain one or more attributes and an attribute can contain one or more values. Finally, values can contain one or more subvalues. Attributes, values, and subvalues are all delimited by special characters: ^ for attributes,] for values, and \ for subvalues.

An item is a string consisting of combinations of these elements and can be up to 32K bytes in length. In PICK BASIC, this string can be loaded into a dynamically dimensioned array.

9.6.1 DYNAMIC ARRAY (FILE) HANDLING FUNCTIONS

Once an item has been loaded into a dynamic array, the EXTRACT function can return the contents of a specific attribute, value, or subvalue. The EXTRACT command then specifies the dynamic array, the attribute number, the value number, and the subvalue number to be extracted. For instance:

```
X = EXTRACT (ARRAY-NAME 5,2,1)
```

This will extract the first subvalue of the second value of the fifth attribute in the specified array. The extracted value is then assigned to variable "X".

The REPLACE function provides the corresponding capability to change the contents of a value in the array. DELETE allows for deletion of a specific attribute, value, or subvalue. INSERT allows the insertion of new attributes, values, or subvalues.

The LOCATE function allows the programmer to specify a location search. The array is searched and the location is returned in a variable. If the string is not found, the value returned is the location the string should be in if a specified sorted order is to be maintained. The program can then insert the string at this location, knowing sorted order will be maintained with no further effort.

The COUNT function will count the number of attributes within an item, the number of values within an attribute, or the number of subvalues within a value.

Using these functions, the full range of data base management ability inherent in the PICK Operating System file management system is available to the BASIC programmer. This not only simplifies the BASIC program itself, but ensures compatibility with the PICK Operating System ACCESS Processor, allowing the development of applications that use BASIC for customized data base management supervision, and the English-like ACCESS language for all associated report generations.

9.7 OTHER FEATURES OF THE PICK OPERATING SYSTEM BASIC

The MATCH statement provides pattern matching facilities in BASIC similar to those available in the PROC processor. These include testing for a number of alpha or numeric characters and literal string comparison.

The CHAIN function will transfer control to another BASIC program or any valid TCL command including a PROC. Variables can be passed to the chained program.

The PRINTER ON, PRINTER OFF, and PRINTER CLOSE statements cause output to be directed to the spooler or the user's terminal. When the program is finished executing, the spooled file will become eligible for printing; if spooling prior to the end of the program is desired, the PRINTER CLOSE statement will immediately spool the accumulated output.

Output masking functions, similar to the same functions in the ACCESS Processor, include justification, flush left, flush right, specification of the number of digits to the right of the decimal point, descaling, suppression of leading zeros, insertion of commas, printing "CR" or the minus sign after negative numbers, printing "DB" after positive numbers, appending dollar signs, and filling a fixed length field with any character.

The HEADING and FOOTING functions, similar to the same functions in the ACCESS Processor, help format pages when output is being prepared for the line printer. A heading or footing can be stored with the related BASIC statement, and the PAGE statement activates the printing of a heading/footing on each page. PAGE can also accept a variable as an argument to set the page number counter. Optional arguments for heading and footing statements will automatically incorporate the time and date, assign page numbers, center text, and insert blank lines.

The PROMPT statement assigns a prompt character to be printed at the CRT whenever the program stops for input from the CRT. The READNEXT statement reads a list of item-ids from a list supplied by an ACCESS Processor SELECT or SSELECT. These items can then be brought into a dynamic array for processing. READNEXT statements can continue until the list is exhausted.

BASIC can access the magnetic tape unit by the READT, WRITET, WEOF (write end-of-file mark), and REWIND statements (refer to Section 12, Magnetic Tape System, for more information).

9.8 MULTI-USER FILE LOCKS

If one or more BASIC programs are running concurrently and they access the same file, multi-user lockout protection is necessary to prevent two programs from writing to the same data without coordination. Even a simple inventory system cannot allow two clerks to run the same inventory program at the same time unless this protection is available. This problem is somewhat compounded in a data base oriented system, since an attribute like "part-number" might exist in one file, but be accessed by many program modules. Without file lockout protection, the entire inventory system might be accessible to only one user at a time. The PICK Operating System BASIC provides a sophisticated set of locks to coordinate multiple user access of the same files.

File locking is implemented with modified versions of the READ and WRITE statements. When one of these modified statements is executed, the group in which the read takes place is "locked" to other programs until released by the locking program. A "group" is an arbitrary subset of a whole file, the modulo* separation.

For the most part, a "group" is a user transparent concept. It is, however, the fundamental block of data the PICK Operating System reads and writes, so it is a handy point in which to implement file security. Since a "group" is a subset of an entire file, two users might still be able to access the file at the same time; they just couldn't access items in the same group at the same time.

If a program attempts to read data from, or write to, a group that is locked by another program, the program will wait until the group becomes unlocked. Use of conditional arguments with the modified READ and WRITE statements can be used to gracefully branch to another part of the program to deal with the situation. The RELEASE statement unlocks groups, and all locked groups locked by a specific BASIC program are unlocked when that program ends. The PICK Operating System can keep track of up to 62 locked groups at a time.

9.9 INPUT AND OUTPUT CONVERSIONS

Some of the input and output conversion specifications intrinsic to the PICK Operating System English-like ACCESS language are also available in BASIC. The conversion codes, stored in attributes 7 and 8 of an attribute definition item, can also be used as arguments in the ICONV and OCONV functions. The conversions applicable to use in BASIC are described below.

9.9.1 D CONVERSION/CORRELATION

D or DATE is used to convert the date from internal to external format. The date is stored in the system as four bytes representing the number of days from a fixed point in time.

9.9.2 MC CONVERSION/CORRELATION

MC or Mask Character is used to convert strings to upper or lower case or to extract alphabetic or numeric characters from a string.

9.9.3 MT CONVERSION/CORRELATION

MT or Mask Time is used to convert the time of day from the internal format to either 12- or 24-hour format.

9.9.4 MX CONVERSION/CORRELATION

MX or Mask Hexadecimal is used to convert ASCII character strings to their hexadecimal (base sixteen) representations, and to convert hexadecimal to ASCII.

9.9.5 P CONVERSION/CORRELATION

P or Pattern Match conversion is used to return only data which matches a specified pattern. The argument can specify a specific number of numeric characters, a specific number of alphabetical characters, or a specific number of either. The argument can also specify a specific pattern as in the following example:

```
P(3N-2N-4N);(9N)
```

This requests a pattern match of three digits, two digits, four digits separated by hyphens (a social security number). This example also illustrates the use of the semicolon in separating multiple arguments. In this case, the second argument of nine digits without hyphens is also acceptable, so both of the following would satisfy as a match:

```
410-96-5664 and 410965664
```

9.9.6 T CONVERSION/CORRELATION

T or Text Extraction is used to extract a specified number of characters from an attribute. The argument consists of the starting character position and the number of contiguous characters to extract.

9.10 MATH FUNCTIONS

The PICK Operating System BASIC contains a number of intrinsic math functions including: square root, random number, sine, cosine, tangent, natural logarithm, exponential, and power.

9.11 SUMMARY

The BASIC processor complements the PICK Operating System with this popular procedural programming language. Since the full-range of data base management functions are available to the BASIC programmer as implemented in the ACCESS Processor, the complimentary combination of capabilities these two processes provide can be used to bring new applications on-line faster than would be the case using conventional file structures and totally procedural languages.

system editor and text page formatter 10

10.1 INTRODUCTION

The Editor is an important, and frequently used, tool on the PICK Operating System. Although an editor is sometimes thought of as a text-processing program, one to be used for creating source listings and documentation manuals, the role of the Editor is somewhat expanded with the PICK Operating System. Using the Editor, any attibute of any item in any file may be examined and altered. While day-to-day data base management will normally occur under the supervision of BASIC programs for input and inquiry, and the English-like ACCESS Processor for report generation, the Editor can be a useful tool for "touching up" the data base, fixing errors, etc. Since the Editor can be controlled by a PROC (refer to Section 8, PROC Language Processor, for more information), simple file alteration routines can be very quickly programmed in PROC.

Runoff, the page formatter, is used in conjunction with the Editor to prepare documentation, manuals, "form letters," etc. for line printer output. One important capability of Runoff is that it can read a second file and insert data from that file into the main file being prepared for printing. This can be used in conjunction with the ACCESS Processor to prepare a list of addresses that meet certain criteria (like 60 days past due) and incorporate those addresses, and other data if desired, into an otherwise static body of text. It is not to be associated with the JET word processing subsystem.

10.2 THE EDITOR

The Editor can be instructed to read an entire file item by item, or a subset of a file using the SELECT and SSELECT ACCESS Processor commands, or the file dictionary. The Editor edits items one at a time. Remember, an item is a string of attributes, separated by attribute marks, starting with attribute 0, the key, which identifies the following attributes. Since attribute 0, the item-id, is used to define the actual location of the file on disk, it cannot be altered by the Editor; thus, editing begins with attribute 1. The Editor numbers each attribute, starting with 001, and each attribute is treated as a separate line on the terminal screen. The attribute marks on disk, ^ , are effectively converted to carriage returns for the purpose of visualizing with the Editor.

In a relatively free-format text file, the significance of an "item" may be flexible and, while editing a file, new items may be inserted anywhere, at any time. An "item" could be a section or a whole manual.

10.2.1 BASIC EDITOR COMMANDS

The Editor "points" to a current line at all times. Several commands control the pointer including: Up, Next, Goto, Top, and Bottom. Some of these also take arguments (i.e., U5 would back up five lines, G121 would go to line 121). List displays the current line. Input allows the user to insert new text between existing lines or, with a new item, to give it a name and start inputting text from line 001 of that new item. The MErge command allows a block of text to be easily moved within an item, or moved from another item or another file into the item being edited. Locate followed by a string argument can be used to search for a string. Restrictions include searching no more than X lines forward and searching only between two listed columns. The column limitation is useful when preparing a program source listing with a label field/statement field structure since the search can be focused on the label itself, or the statement in which it is referenced. The Again command repeats the previous search command, searching for all occurrences of a given string. The Replace command performs a search operation and replaces the search string with a specified replace string. The DElete command will delete specified lines or, in a manner similar to search, delete lines which contain a specified string in them. The X command will reverse the effect of any previous command, so if an error was made, the user can type X to restore things to their former state. The Prestore command will store a sequence of Editor commands to be invoked later with one keystroke.

10.2.2 SPECIAL EDITOR COMMANDS

For assembly language programming, the Editor has a special columnar function to assist in formatting source code. Normal assembler programs include a hexadecimal display of the object binary code generated by the assembler in the first field. This can be suppressed, if desired. The columnar restrictions in the locate command, described above, are also useful when preparing program listings.

10.3 FORMATTER

Runoff is a formatter used to process text files for output to a printer. The Runoff command vocabulary is inserted into the text stream and, as Runoff processes this source file, it reads commands sequentially and creates an output version accord to the embedded instructions. Runoff commands appear on a line by themselves, each command starting with a period. Thus, any line which begins with a period is assumed to contain one or more commands. Runoff can justify text, print in bold face, and control pagination, running headings and footings, etc. When Runoff output is directed to a CRT terminal, commands for boldface and underlining may be temporarily suppressed.

10.4 SOME BASIC RUNOFF COMMANDS

10.4.1 PAGE FORMAT

The PAPER LENGTH n command sets the number of lines to be printed on a page to n. LEFT MARGIN n causes the margin to be moved to the right n spaces. LINE LENGTH n sets the printing line length used in the FILL and JUSTIFY commands. The length starts from the left margin. The SPACE, SPACING, and SKIP commands can be used to control single or double spacing and to create blank lines in a document as required. The FILL mode causes the source text to be formed into lines which are as long as possible, given the chosen line length. The JUSTIFY mode causes filled lines to be padded with randomly inserted spaces so they are all exactly as long as the chosen line length. The CENTER command causes the next line to be centered rather than filled or justified. The CAPITALIZE SENTENCES mode forces the first letter following a ., ?, or ! to be capitalized. This mode will also force a double wordspace after a ., ?, :, or ; unless that character appears at the end of a line. The BEGIN PAGE command breaks the page at the present line. The TEST PAGE n breaks the page only if there are less than n lines left. This conditional command can be used to hold charts, tables, captions, and related text together on the same page. PARAGRAPH n is used to control the formatting of paragraphs. The argument n controls positive or negative indenting. Paragraphs can be identified by leaving blank lines between them, or starting the first line with one or more blank spaces. The INDENT n command causes the following text to be indented n spaces from the margin. A negative number may be used for "hanging" indents.

10.4.2 DOCUMENT STYLE

The CHAPTER n command creates a beginning of chapter page taking the argument n as the title. CHAPTER also assigns chapter numbers sequentially and accumulates chapter starting point page numbers for the table-of-contents processor. SECTION n "text" is used to sequentially number paragraphs within a chapter. The sectional sequence is determined by a number n. The text following the section number, "text," is printed as a paragraph topic heading. Like the CHAPTER command, each topic heading and paragraph number is passed on to the print table-of-contents processor. The FOOTING text command has special arguments, which appear in quotes, and can be inserted anywhere in text. These include P which will output the current page number, L which forces a blank line, I which prints out the current item-id, D which prints the date only, and C which causes a given line to be centered. The HEADING text command functions exactly like FOOTING except the text is output at the top of every page.

10.4.3 INDEXING

INDEX text enters the item text into the index under the page number the item appears on. The index references are accumulated as pages are built by Runoff and passed on to the printer index processor. PRINT contents prints a table of contents based on the accumulated statistics from the chapter and section commands after Runoff is finished producing the main body of the document. PRINT index prints an index of topics and page numbers based on the accumulated index references. The index is produced in two-column format.

10.4.4 GRAPHIC DEVICES

BOX m,n causes the following text to be surrounded by a box of the width n and margin m. Text is "boxed" until a BOX command without the m,n arguments is reached. The HILITE c and HILITE OFF commands can be used to flag sections of the text for review, to call attention to revisions, etc. The argument c is the character that will be printed at the extreme right margin for every line until the HILITE OFF command.

10.4.5 MISCELLANEOUS COMMANDS

Most of the commands discussed above have related opposites which negate their effect (i.e., NOJUSTIFY which turns off the effects of the JUSTIFY command). Other commands direct output to the user's CRT or to a line printer, allow boldface and underline printing, change tab settings, etc.

10.4.6 SPECIAL COMMANDS

A number of functions are available to access data in other files during Runoff. These can be used to address form letters from a sorted address list stored in another file, or to totally customize a series of letters that access account balances, past due amounts, etc., as well as a basic name and address.

The INPUT command causes the Runoff process to stop for a line of text to be input from the user's terminal before proceeding to process the rest of the stored file. The CHAIN new file command at the end of a file causes Runoff to begin reading a new file. This can be used to "gang" a group of files into one printout, as in a chapter oriented document where each chapter is logically a separate file. Since Runoff formats a file brought in by the CHAIN command using the same parameters that have been accumulated from the previous file, only the first file needs to have parameters like line length, margins, header definitions, etc. CHAIN can also chain to the same file and form a loop. In conjunction with the READNEXT command described below, this feature can be used to reprint the "same" letter to different addresses on a pre-selected list. The READ item-id command causes Runoff to use item-id as the source text until the contents of the item-id are exhausted, at which point control is then passed back to the "host" file. Optional arguments specify which file is to be read unless the item-id is in the same file as the "host" text. The READNEXT command is used in conjunction with the ACCESS Processor functions SELECT and SSELECT which prepare a list of items as a subset of a file (refer to Section 4, Data Base Architecture and ACCESS Process, for more information on the capabilities of SELECT and SSELECT). READNEXT reads attributes one at a time from this preselected list. This command can be used to insert unique data into an otherwise standard body of text. The CHAIN command, described earlier, can force Runoff to "loop back" for another pass through the file, where the READNEXT commands will indeed read the next attribute from the pre-selected list. This process will continue until the list is exhausted and the Runoff program ends.

10.5 SUMMARY

Both the Editor and Runoff provide powerful tools that enhance operations of the PICK Operating System. Aside from the manipulation of data base files, the Editor is used to input source statements for Assembler, BASIC, and PROC programs. The Editor, in conjunction with the formatter Runoff, provides text processing tools for the preparation of lengthy documents and, in conjunction with the data retrieval facilities of the ACCESS Processor, can prepare "form letters" which include information fields taken from other files.

print spooling system 11

11.1 PURPOSE OF PRINT SPOOLING

Unlike much of the computer system, the printer can only service one user at a time, and since a printer is often used to print on a variety of different forms which must be manually changed, there is often a conflict between the computer's ability to generate large amounts of output and the bottleneck which can accumulate at the line printer. The sophisticated line printer spooling system provided with the PICK Operating System is designed to make the printing function as efficient as possible and performs two basic functions to this end. It allows data which is destined for the printer to be written to disk or tape immediately as it is created, regardless of whether a printer is available at the time, and it allows the printer or printers to be properly managed by the personnel responsible for them.

11.2 LINE PRINTER SPOOLER

The PICK Operating System spooler software is an elaborate, flexible, yet easy-to-use system for the generation, management, and printing of reports. The spooler will accommodate up to 125 different queues, up to 16 serial printers, and 600 individual files. The queue structure allows print requests that call for special forms to be sent to different queues set up for those forms. Personnel in charge of operating a high-speed printer may examine the various queues, mount the necessary forms, and print all the files in those queues before going on to the next form. Special queues can also be established for high- or low-priority jobs, high- or low-speed printers, and local or remote printers. An individual printer can be assigned to "work for" up to three different queues and a print file can be moved from queue to queue at will.

An individual file can be "spooled" to an active queue and printed on a next-in-line basis, to a hold file to be dealt with later, or directly to an available printer or the magnetic tape drive. The first 500 bytes of a file can be examined, alignment can be checked before beginning a long run, and aborted runs can be restarted on any page. All commands which allow the inspection of print files or the modification of their status are security protected, and typically only function for files from the user's own account, or for the system operator.

The number of pages spooled is recorded in the accounting history file for each user.

11.3 SPOOLER COMMAND VOCABULARY

The following commands are the most important and frequently used of the spooler commands.

11.3.1 DESTINATION COMMANDS

Files which are to be printed can be given several destinations. Each user is "assigned" printer 0 when he logs on and this is the "default" condition. This can be changed at any time. Files may be created on disk and flagged to be printed as soon as a printer is available, held for operator intervention before printing, or the file can be sent to tape instead of disk. A print file can be automatically deleted upon completion of printing, or held for manual deletion. Combinations of these can also be performed. For instance, a print file can be sent to tape and disk. The disk version could be used to print, when convenient, and the tape version placed in the archive.

For extremely large print runs, another procedure is useful. A file can be immediately printed; as it is being created, a choke mode is used to stall the creating process when it gets 20 disk frames ahead of the printer. This prevents more than 20 frames of disk space to be used in the storage of large print files, while allowing several pages of "buffer" for a restart in the event the printer fails.

11.3.2 FORMS CONTROL

The form number specification routes a print file to a printer which services that form. Up to 125 forms or "queues" can be handled by the spooler and a printer may service up to three of these queues. The align function allows the operator to print a specified number of lines to test form alignment before starting the run. A queue does not necessarily have to represent a form. Certain queues might be reserved for rush jobs or low-priority overnight service.

11.3.3 MULTIPLE COPIES

The copy count specification indicates how many copies are to be printed, up to a maximum of 125.

11.3.4 EDITING PRINT FILES

Files in the spooler system, whether held for printing on a next-in-line basis, held after printing, or just waiting for operator intervention, can be "edited" using the edit options. They can be deleted, moved to another queue, written to tape, the first 500 characters can be examined, the number of copies altered, etc. Most of the commands will act on logical groups of files. For instance, if the accounting department generated several reports, all accounting reports could be dispatched to the printer with one command.

11.3.5 RESTARTING ABORTED RUNS

The STRING function allows the spooler to search through a file and find a unique string of characters. By answering this prompt, a file aborted because of a paper jam, broken ribbon, etc. can be restarted by searching for an appropriate restart point. The spooler will put the beginning of file marker at this location and begin printing. Form alignment will occur automatically on the next page.

11.3.6 PHYSICAL PRINTER MANAGEMENT

Several commands are available for the use of personnel actually involved in setting up printers, change forms, etc.

11.3.7 PRINTER STARTUP

The STARTPTR function is used to bring a printer "on-line" and tell the spooler it is available for use. Start printer allows the operator to assign the printer to the queue or queues that match the form which is mounted on the printer. Certain other parameters, such as the number of pages to skip between jobs, the type of printer, etc., are assigned at this point.

11.3.8 PRINTER HALT

The STOPPTR and SP-KILL functions are used to delete printers from the system. STOP can be used for form changes, ribbon changes, or other maintenance. STOP halts a printer at the end of the current job; KILL halts the printer immediately in the event of jamming or other malfunction.

11.3.9 SPOOLER STATUS

The LISTPEQS function displays the files which are in the spooler, the accounts they came from, status information (printed, holding, etc.), the length of each file, the number of copies to be printed, the form on which they are to be printed, the time and date they were spooled, etc. Specific inquiries can also be made, such as what files are waiting for a particular form.

magnetic tape system 12

12.1 INTRODUCTION

The magnetic tape drive subsystem is a integral part of the PICK Operating System computer. Magnetic tape is an ideal medium upon which to archive large but seldom used files and to provide "back-up" of valuable data in the event of a system malfunction.

Total security requires that a duplicate set of back-up files be maintained off the main computer site as a precaution in the event of fire or other massive destruction. Increasing pressure from government regulatory agencies mandates preservation of records for years. In either case, the amount of data involved in a back-up program is large, and the convenience and cost of the secondary storage system becomes a valid concern. Magnetic tape is, byte-for-byte, the most inexpensive storage medium available and operation of the tape system does not interfere with other processes on the system.

Because the PICK Operating System is so often used for data-base intensive applications, the probability of a very large amount of important and volatile data being on-line makes a well designed, convenient magnetic tape system a must. The PICK Operating System tape handling system is a well integrated part of the total computing environment. Over 15 commands are available for control of this system, including many utility functions like ASCII to EBCDIC conversions. Only the most significant functions are discussed here.

12.2 ACCESSING THE TAPE SYSTEM

Since the tape system cannot be used by more than one user at a time, two verbs are provided to control the status of the tape drive.

12.2.1 T-ATT AND T-DET VERBS

The T-ATT, or tape attach, verb "attaches" the tape drive to the user's job and shuts out all other users until the T-DET, or tape detach, verb is executed. Should the tape drive already be attached to someone else's job, an error message to that effect is issued.

Optional arguments for the attach verb give the record length (variable between 80 and 8000 characters) that will be used for the session.

12.2.2 TAPE HANDLING VERBS

1. T-FWD and T-BCK provide forward and backward movement one or more records at a time.

2. T-REW rewinds the tape to the Beginning-of-Tape (BOT) mark.

3. T-SPACE n moves the tape past the tape label area and on the End-of-File mark n times.

4. T-EOD (end of data) moves the tape forward to the EOF mark at the end of the last file on the tape.

5. T-WEOF writes an EOF mark on the tape.

6. T-READ reads from the tape to either the terminal or line printer.

7. SP-TAPEOUT reads from the tape and passes the file into the line printer spooling system for ultimate disposition.

12.2.3 T-DUMP, S-DUMP, AND T-LOAD VERBS

The T-DUMP, S-DUMP, and T-LOAD verbs are ACCESS Processor verbs which provide an access method to the tape system with a close relationship to the data base file structure. T-DUMP and T-LOAD copies files to and from the tape system, s-dump provides a selection capability similar in form to the ACCESS Processor "SELECT" command. For instance, if a mailing list were to be transferred to tape, the selection process could read the entire file, but only transfer those names with the zip code between two values.

12.2.4 TAPE LABELS

The tape system can label each reel of tape with an 80-character label. This label normally includes the file name and a user-supplied heading plus the record length, time and date, and the reel number. The T-RDLBL verb can be used to read the label of a tape; however, most tape processes automatically create and write, or read the tape labels when appropriate. For instance, when reading from a multi-reel volume, the system prompts with a mount for the next reel automatically.

12.3 SUMMARY

The PICK Operating System magnetic tape handling facilities are a well-integrated part of the system, and provide for off-loading data, backup, and transporting data from one computer to another.

COMPU-SHEET and ACCU- PLOT 13

13.1 COMPU-SHEET

COMPU-SHEET is an electronic worksheet which allows you to use the computer to perform complex business analysis and print out reports. Since COMPU-SHEET can retrieve data from any file in your system and use it for display and calculation, it provides the ability to solve problems which may take hours to accomplish by hand.

The many uses of COMPU-SHEET include cash flow and other financial projections, budget analysis, job costing, estimating, advertising analysis, planning, proposals, investment analysis, sales forecasting, modeling and trust funds.

In effect, COMPU-SHEET gives you a large blank sheet of "paper" where you can define and solve your business problems. The sheet is divided into columns and rows. You choose the number of columns and rows, the width of each column, and specify how the data should be displayed. You may begin at any location, define headings and start to enter your information. A location may contain any type of data (alphabetic or numeric) and may also contain a formula to operate on data in that location or on data in any number of other locations.

There is no limit to the number of columns or rows you may use. Your screen will scroll up and down and from side to side to let you create a much larger worksheet than can appear between the limits of one screen. If you think of your screen as a window which displays only a portion of a much larger worksheet, it is easy to visualize how this window may be moved around to uncover any portion of the worksheet. Later, you may print out your worksheet in its entirety.

13.2 ACCU-PLOT

ACCU-PLOT is a user-friendly graphics software package that operates under the PICK operating system to produce quality graphics. You can create bar charts, line charts, scatter-graph diagrams and pie charts.

To use ACCU-PLOT, you construct a single ACCESS sentence which contains a special ACCESS verb followed by the filename of the file that contains the data to be graphed. The sentence may include selection-criteria, sort-criteria, print-limiters, output specifications, headings and footings. Multiple value attributes are acceptable and any number of attributes may be plotted.

ACCU-PLOT interfaces with the BASIC processor so that you may include graphic representations in your BASIC programs.

ACCU-PLOT also interfaces with COMPU-SHEET to allow you to represent your worksheets as line graphs, bar graphs, pie charts, etc. Also, if you have a color terminal, you can display your graphics in color.

JET word processor 14

JET is the PICK operating system word processor. It provides a full range of word processing operations including document creation, format definition, editing, and printing.

JET is implemented through a CRT terminal whose keyboard contains the conventional typewriter character set as well as special function keys. The CRT video screen displays the text during creation and editing essentially as it will appear when printed. (There is an option to suppress or display format and print control commands.) JET distinguishes between two types of documentation:

1. Continuous line text; such as, books, reports, letters.
2. Single line text; such as, computer programs and lists of various kinds.

Continuous line documentation is handled by the JET-IN operating mode, single line text is produced by the JET-EDIT operating mode, while printing of both types of documentation is accomplished through JET-OUT operation.

The full text operations carried out by JET-IN are broken down into three further modes: RULER, INPUT and EDIT. Passage from one operating mode to another and to the various modes within an operating mode is accomplished by MODE TRANSITION commands.

All JET-IN and JET-EDIT functions are made easily accessible by use of screen-displayed menus which identify the terminal keys to be used for each function within the particular mode.

Terminal keys perform a multitude of functions. Not only do they create text, define formats and perform editing tasks, but they are also used for cursor control, mode entry and edit, and for viewing the various mode menus that tell which keys do all of the above.

JET-IN mode text creation follows the same procedures you would follow when creating a document on a typewriter. The three steps involved are:

1. Defining the format and organization of the document.
2. Creating the document.
3. Editing and modifying the document.

The JET-IN RULER mode handles the format and organization of the document.

The RULER mode provides screen-displayed menus that include a ruler scale and a ruler template so that you may easily define the format of your document. Definition functions include: margin positions, tab settings, title and heading locations, end-of-line hyphen and automatic word-wrap zones.

Once a text ruler is defined (the definition of the format for a document is called a "text ruler"), it can be saved in a text ruler menu or embedded in the text it has defined. Up to 9 text rulers may be saved for subsequent display, review and possible modification in the RULER mode menu. An unlimited number of text rulers may be embedded in a document's text. These may also be viewed and modified if desired.

The JET-IN INPUT mode handles full text creation (continuous sentences and paragraphs as opposed to line-by-line listings).

Most of this is accomplished by simply typing in the text on the terminal keyboard. As the text is created, it will appear on the CRT screen. A CRT position indicator called a "cursor" identifies where the next character that is typed in will be located on the screen. After the character is input, the cursor automatically advances to the next character position. Other text creation features handled by JET-IN INPUT mode include page numbering, page headings and footings, hilighting, boldface, and underlining.

JET-IN EDIT mode allows you to modify the text created in JET-IN INPUT mode. Editing functions include character deletion, replacement and transposition, word deletion and replacement, line deletion and insertion, sentence deletion, upper and lowercase changing, searching and replacing, spelling checking, and a text move feature called "cut and paste."

The JET-IN INPUT mode for single line text creation operates in the same manner as JET-IN INPUT for full text documents. However, JET-EDIT INPUT does not implement page numbering, headings and footings, hilighting, boldface, margin justification, line spacing or underlining.

JET-EDIT EDIT mode lets you modify the text created under JET-EDIT INPUT mode. Most of the editing features with the commands that were used by JET-IN EDIT are also used by JET-EDIT EDIT.

JET-OUT provides for the printing of both JET-IN and JET-EDIT created texts. In addition, the Utility functions from both INPUT modes allow JET-OUT to automatically insert text from other sources into the text it is printing. These Utility functions include: append other text, insert other text, insert item name, insert attribute values, insert text during printing, and halt printer output.

Exiting from one mode and entering another, called "Mode Transition," is accomplished by pressing certain specified keys. The screen-displayed Mode Menus include this information, so there is never a question of what to do next.