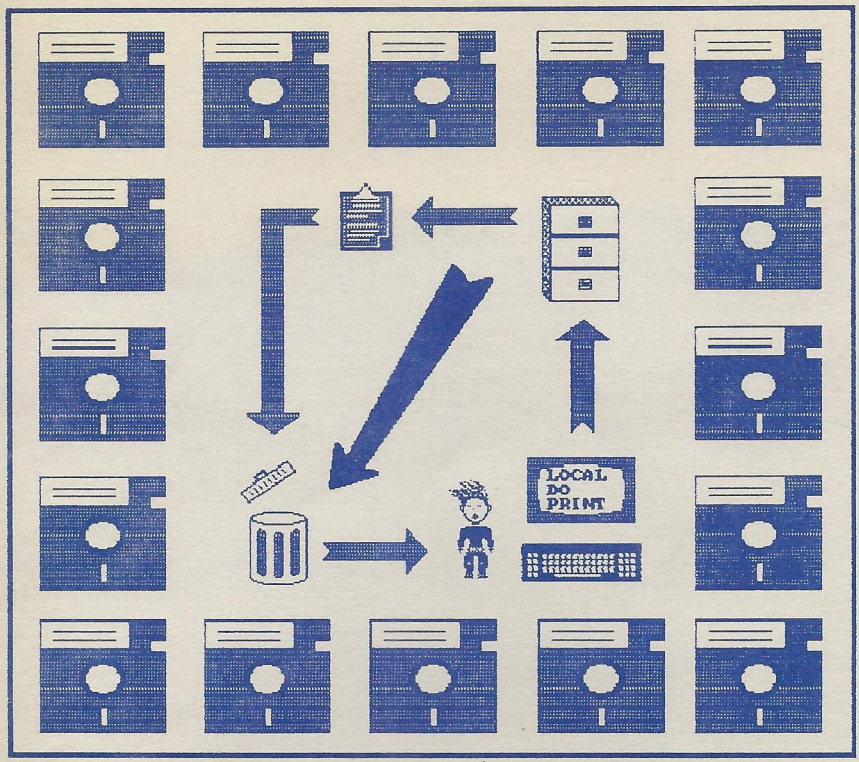
TI-BASE



FOR ALL YOUR DATA NEEDS

This document describes the capabilities of TI-BASE version 2.0

LIABILITIES

INSCEBOT will not be responsible for any losses resulting from the use of the products described in this document. Actual liability will be limited to the purchase price.

LICENSING

This product may not be duplicated for resale or distribution without the express approval of INSCEBOT Inc.

WARRANTY

INSCEBOT warranties this product for a period of 90 days from date of purchase. Replacement will be accompliahed by return mail at no cost when the original diskette is returned. After 90 days, there will be a $5.00 replacement fee.

Upgrades to the most current versions of TI-BASE will be provided at a cost of $7.95 including postage, and the return of the original diskette.

Contents

Section 1
INTRODUCTION

Section 2
GETTING STARTED

Section 3
OPERATING INSTRUCTIONS

SECTION 1

INTRODUCTION

1.1 OVERVIEW

Welcome to TI-BASE. As you work through this manual and the TI-BASE tutorial you will find that TI-BASE is unlike any other data management program available for the 99/4A computer. With its command file programming language, customizable printer drivers, ability to perform manipulation of mathematical data and the host of other features you will read about in succeeding pages, TI-BASE is quite simply the most flexible data management system available for the TI-99/4A computer.

Because we felt that the TI community needed a true, full-featured data base management application, we have tried to provide a flexible tool that allows data to be described and stored in a manner which is easy to use and easy to access. If you are currently using a data base management program on your TI, chances are you can convert existing files to TI-BASE format with no re-keying of data. Once you have the files in TI-BASE format you have the ability to do more "with" the data than ever before. We hope that the TI-BASE program proves to be a useful addition to your software library.

1.2 HARDWARE

TI-BASE is designed to operate on a TI-99/4A home computer with 32K memory expansion, at least one disk drive and any of the following loading environments:

- Editor/Assembler module.
- Extended Basic module.
- Mini-Memory module.

TI-BASE has been successfully tested with the following hardware devices:

- o TI Floppy disk controller
- o Cor-Comp Floppy disk controller
- o MYARC Floppy disk controller
- o MYARC Ram-disk
- o Horizon Ram-disk

1.3 DELIVERY MEDIUM

The TI-BASE package contains the program diskette, the TUTOR diskette, this manual and a keyboard overlay. All diskettes are in SS/SD format. TI-BASE is not copy protected.

1.4 TERMINOLOGY

Append	to add to the end of, as in appending a new record to an existing data base.
Boolean Operator	the same as relational operators such as less than, greater than, equal to etc.
Command File	a collection of commands and/or directives created to perform specific and often repetitive data management functions.
Concatenation	the process of joining two or more pieces of data into one piece of data.
Data-Base	a collection of records.
Driver	a set of parameters within a file that are created for a specific piece of hardware.
Dot Prompt	the indicator that TI-BASE is ready to accept input.
Field	the smallest element of a record.
File	a collection of records.
Heading	the group of names that include the fields that you define in your data base.
Interactive	a mode of operation where the program expects input from the operator.
Literal	a constant value such as PI, which is not changeable.

1.4 TERMINOLOGY (cont'd)

Local	a user-defineable variable used to represent some character, date or numeric value.
Logical Operator	ANDs, ORs and NOTs used to provide or establish some logical relationship between differing pieces of data in a data base.
Nested sort	a term indicating the ability to sort a file by more than one field or a file that has been sorted by more than one field.
Record	a collection of data entered into fields.
SETUP File	the first file read by TI-BASE after the current date has been entered.
Slot	a file hierarchy indicator or position. TI-BASE supports slots 1-5.
Status	the current state or settings of the variables that control many TI-BASE defaults.

SECTION 2

GETTING STARTED

2.1 BEFORE YOU BEGIN

Make a backup copy of the TI-BASE program and TUTOR diskettes before loading the program or processing any data. Any disk copier may be used. Store the original diskettes in a safe place.

2.2 THE TUTORIAL FILES

Included with TI-BASE, is a second diskette called TUTOR. This disk contains files which describe many of the TI-BASE functions. To use the TUTOR files TI-BASE must be loaded. TUTOR is activated by placing the TUTOR disk in drive #2, setting the DATDISK to DSK2 and entering the directive "DO TUTOR".

Once TUTOR has been invoked, a short narrative display of TI-BASE functions appears on screen. These displays are broken into sections, any of which may be invoked separately.

The organization of the TUTOR command file may be examined by listing the file with a "DO TUTOR LIST" directive.

TUTOR also includes sample command files which serve to illustrate the usage of TI-BASE directives. A README/DOC file resides on the TUTOR disk to describe the disk contents.

2.3 LOADING PROCEDURE

TI-BASE may be loaded using the Editor/Assembler, Extended Basic or Mini-Memory modules. Loaders are provided for the following environments;

- o Load and Run (E/A option 3, Mini-Mem) Enter DSK#.TIBASE
- o Program Image (E/A option 5) Enter DSK#.TIBASEP
- o Extended Basic environment
  Automatically uses the DSK1.LOAD file which will run TIBASEB file.

The disk drive # entered in any of the supported loading environments may be drives 1-9.

2.3 LOADING PROCEDURE (cont'd)

When TI-BASE is in the process of loading, multiple files are being read into memory. The drive number to load from, after each loader is entered, is determined by the value that is contained in VDP memory, at address >3FF5. This is the standard address used by TI. Since some hardware devices do not follow this standard there may be times when TI-BASE is unable to determine the drive to use. In those cases a prompt will appear requiring the user to input a single numeric digit (1,2,3 etc.) defining the drive to use. The prompt will appear at the upper left of the screen as a;

"enter drive #"

statement. If you are trying to load TI-BASE from DSK1 you would simply press the 1 key to continue loading. Thereafter, DSK1 will be used as the access drive for the remaining program segments.

Editor/Assembler

Select Editor/Assembler from main menu. Place TI-BASE disk into disk drive (1-9).

Either

Select option 3, "Load and Run". Enter "DSKn.TIBASE" where n is the drive # (1-9).

OR

Select option 5, "Program Image". Enter DSKn.TIBASEP" where n is the drive # (1-9).

Mini-Memoy

Select Mini-Memory from main menu. Place TI-BASE disk into disk drive (1-9). Select option 3, "Re-initialize". Depress "Function Proceed" (F6). Select option 1, "Load and Run". Enter "DISK#.TIBASE" where # is the drive # (1-9.

Extended Basic

Place TI-BASE disk in drive #1. Select Extended Basic from the main menu.

John Johnson Menu from Horizon Ram Disk

Select option 3, RUN A PROGRAM, and then type in DSK#.TIBASEP

2.4 INITIAL LOADING

When TI-BASE has loaded the prompt;

"enter date (mm/dd/yy)"

is displayed.

Respond by entering the date in the defined format. The date that you enter is appended to any file that is created and/or changed during the time that TI-BASE remains in memory. The value for the date that you enter may be used in TI-BASE command file programming or in other areas as the .DATE. variable.

Once the date is entered the system will look for and execute a file named "SETUP" on the drive from which TI-BASE was loaded. The "SETUP" file is to TI-BASE as a program named LOAD is to Extended Basic. That means that TI-BASE will execute whatever instructions it finds in the SETUP file.

When the SETUP file instructions have been executed a dot prompt (the period symbol) is displayed and the system is ready to accept TI-BASE directives.

2.5 ON-LINE HELP

There are several help files available for use while TI-BASE is active. They are invoked by pressing the Fctn HELP (F7) key.

If HELP is invoked when you are in any of the program's interactive editors (APPEND, EDIT etc.), the function key options will be presented. ESCAPE (F9) will return the program to the editor that you were in when the HELP key was pressed.

If HELP is invoked when you are at the dot prompt, a hierarchy of HELP screens is available. Access to a desired HELP screen is made by pressing the digit key associated with the desired HELP function. Pressing <ENTER> will return to the previous help screen while ESCAPE (F9) will return to the dot prompt.

All HELP files are looked for on the device defined by the system status in the "PRGDISK" variable. This means that the program disk, which contains the HELP files, must be in whatever drive you have listed in the "PRGDISK" value in the SETUP file.

2.6 SETUP FILE

When the system is first loaded, a command file named "SETUP" will be executed on the device from which the system was loaded (usually DSK1). This allows the system defaults to be initialized automatically each time TI-BASE is used. It also allows any constants desired as local variables to be defined and it allows you to specify other instructions such as the automatic loading of a data file, the automatic execution of a command file or the automatic choice of a printer driver.

Modifications to the SETUP file may be accomplished by issuing the command MODIFY COMMAND SETUP from the dot prompt.

Since the MODIFY COMMAND directive looks to the disk drive specified for data files (because the data disk is where command files are saved) you will need to remember to either include the device in the directive, or you can simply set the DATDISK variable to the same drive that the PRGDISK uses (usually DSK1). This last option would be accomplished by typing in;

```
SET DATDISK=DSK1.
```

from the dot prompt.

2.7 THE STATUS or INFO LINE

The bottom of TI-BASE's opening screen contains an inverse video bar that displays various bits of information about what's going on in the system. The bar is 40 columns wide with various segments of the bar dedicated to specific information. The chart below explains each segment by position, with position number 1 being the left-most location on the bar.

Pos#	Operation/Function
01-08	The name of the Command file currently in use.
09-11	The Command File line number being executed.
12-	Blank.
13-	The slot number that is currently active.
14-	Blank.
15-22	The name of the active data-base.
23-27	The number of the record being read/written.

2.7 THE STATUS or INFO LINE (cont'd)

28-	A slash separater between the active record and the file size.
29-33	The file size in number of records.
34-36	End of file indicator.
37-39	Insert mode toggle indicator.
40-	Operation indicator; *-Pause, C-close, O-open, R-read, S-status, W-write, D-delete

2.8 COMMANDS VERSUS DIRECTIVES

Except when single function instructions are issued from the dot prompt, TI-BASE uses "directives" within "commands" to accomplish data manipulation. The words;

```
FIND, DISPLAY, PRINT, REPLACE
```

etcetera are directives.

The phrase;

```
DISPLAY ALL SUBJECT SOURCE TYPE DATE PAGE
```

is a directive. Command Files, are simply collections of specific directives created in desired order.

SECTION 3

OPERATING INSTRUCTIONS

In this section, we have tried to group common functions to give a cohesive view of TI-BASE operations. Individual directive descriptions may be found in section 4 of this document.

For demonstration purposes, we will refer to a data-base named STOCK, which will contain descriptive items which might be found in a warehouse. A copy of this data-base with some sample records may be found on the TUTOR disk.

Single disk users may obtain optimum performance by copying file OVRLAY/P onto each of your data disks. PRGDISK and DATDISK should then be set to the data disk device using the SET command. This will give you all of the capability described in this document except the FORMAT and disk to disk COPY. If help screens are desired, the files beginning with AID... may also be copied onto the data disk.

3.1 STATUS

The system status provides a display of default values used in the system. These values define such things as default devices, printer definitions, and switches used by the system.

DATDISK Defines the default device to be used for data-base files and command files.

PRGDISK Defines the default device to be used for program files and help files.

PRINTER Defines the printer definition to use when the PRINT, LIST, or SNAP directive is issued.

PAGE Defines the number of lines per printer page.

HEADING ON/OFF....Determines if the field headings will be output during display and print operations.

TALK ON/OFF Determines if the command directives will be displayed as they are executed.

3.1 STATUS (cont'd)

SPACES Defines the number of spaces to place between fields during display and print operations

RECNUM ON/OFF.... Determines if the record number will be output during display and print operations.

LSPACE Defines the amount of dynamic space to be allocated for local variables.

CURSOR Allows the cursor speed to be altered. For my tastes, 2 works well on the TI-99 and 10 is adequate for the GENEVE.

DATE Current date.

3.1.1 DISPLAY STATUS

The values associated with system status may be viewed with the DISPLAY STATUS directive. The result of this directive is a display of the current values on the console.

3.1.2 DEFAULT VALUES

```
DATDISK = (device from which TI-BASE was loaded)
PRGDISK = (device from which TI-BASE was loaded)
PRINTER = PIO.CR.LF
PAGE    = 56
HEADING = ON
TALK    = ON
SPACES  = 1
RECNUM  = ON
LSPACE  = 256
CURSOR  = 10
DATE    = (date from date prompt)
```

3.1.3 CHANGING STATUS PARAMETERS

The values associated with a status parameter may be changed using the SET directive.

i.e. SET TALK=OFF

changes the value of status parameter TALK to OFF

3.2 DATA-BASES

TI-BASE is designed to permit the user to define a wide variety of customized data-bases Once defined, there are provisions to enter data interactively or programatically.

- o Up to 5 data-bases active at any one time
- o Data may be freely interchanged and manipulated between the current records of any active data-bases
- o Full mathematical capability.
- o Numeric and character literals.
- o Nested sort capability.
- o Ability to "FIND" related data.
- o Global data-base processing.
- o Display and Print capability.
- o Command language with structured directives.
- o Local variables.

3.2.1 CREATING A DATA-BASE

Creating a data-base consists of defining the content of each record in terms of names, data-types and sizes. This allows TI-BASE to create a data-base file and be able to associate the names that you define with the values that you enter. The data-base will be a collection of the records you create. The following data-types may used in creating a data-base:

(C)haracter -- The width specifies the number of characters to be contained in the field.

(N)umeric -- The width specifies the total number of digits including sign and decimal. Note that a sign is mandatory which means the minimum size is 2 for a numeric type. The decimals specify how many digits are placed to the right of the decimal point.

(D)ate -- The width is always 8 which includes the slashes between month/day/year.

(X).-- The width must be even. Values are entered as HEX digits and represent control codes to be transmitted to the printer.

New data-bases may be defined with the CREATE directive as follows:

CREATE DSK2.STORE

The above directive will allow creation of a data-base named STORE on device DSK2. Data-base names may be up to 8 characters long. If the device is omitted, the default device in DATDISK will be used.

There must be a formatted diskette in the selected device.

Once the directive has been entered, an interactive display will allow the definition of the data-base structure. You may enter up to 17 fields of (C)haracter, (N)umeric, (D)ate, or (X)printer control, data types. As an example, suppose you were to define a warehouse data-base. The structure might be as follows:

Fieldname	Type	Width	Decimals
ITEM	C	20	
STOCK#	C	10	
LOCATION	C	15	
QUANTITY	N	6	
WHOLSALE	N	8	2
RETAIL	N	8	2
MFR	C	20	

The above example would create a data-base with 7 fields defining items which might be contained in a warehouse.

Once the structure is defined, EXECUTE (F8) will continue the process of creating the data-base. At the completion of this, you will be asked if you wish to input data at this time. If so, an interactive display will allow data to be entered. If not, the data-base will simply be available on the named device for later use.

Once a data-base exists and is in use, the structure may be modified with the MODIFY STRUCTURE directive. The current capabilities allow the names to be modified with no impact. If the record size is changed, existing data will be deleted.

The number of data-bases you may have available is limited by the devices you have at your disposal. Any 5 data-bases may be active at any one time. Each data-base may be active on a different device if desired.

Data-base limitations are:

255 characters per field.

17 fields per record.

16129 records per data-bases

3.2.2 USING A DATA-BASE

The USE directive causes an inactive data-base to become active in the current slot.

i.e. USE DSK2.STOCK

Will cause the data-base named STOCK to be accessed and positioned at the first logical record in the current slot. If the current slot already has a data-base active, an error message will result. If the device is omitted, the default device specified in DATDISK will be used.

Once the data-base has been used, data may be edited, appended, replaced, displayed and/or printed as desired. If multiple data-bases are to be active, select another slot with the SELECT directive and USE the next data-bases

3.2.3 DISPLAY STRUCTURE

Once a data-base is active, the structure may be displayed by the DISPLAY STRUCTURE directive. This displays the structure of the current data-base in the current slot.

3.2.4 APPENDING DATA

Once a data-base has been used, data may be entered by the APPEND directive. APPEND will invoke an interactive editor which will allow data to be appended.

3.2.4 APPENDING DATA (cont'd)

As each record is filled with data, either an EXECUTE (F8) or an ENTER at the last field will cause the data to be stored and another blank record be made available for appending data.

When the APPEND session is complete, an ESCAPE (F9) will return to the main screen.

APPEND BLANK will append a blank record to the current data-base and return immediately. This allows blank records to be appended in the command language and data to be replaced into the records programatically.

3.2.5 EDITING DATA

Existing data may be modified using the EDIT directive.

```
EDIT 5
```

Invokes an interactive editor and presents record 5 of the current data-base for editing. If the record number is omitted, the current record will be edited.

While in the interactive editor, you may page forward and backward through the current data-base using function keys F5 and F6.

When you have finished editing data in a record, the data may be saved by EXECUTE (F8) or by depressing ENTER at the last field in the record.

When editing is complete, ESCAPE (F9) will return to the main menu.

3.2.6 DELETING DATA-BASES

A data-base may be deleted from the storage medium by first using the data-base and then deleting it as follows:

```
USE DSK2.STOCK
DELETE DATABASE
```

The above sequence of directives will permanently delete a data-base named STOCK from DSK2.

3.2.7 DELETING RECORDS

Deleting a record from a data-base is a two step procedure. The record must first be marked for deletion as follows:

```
DELETE RECORD 5
```

The above directive will mark record 5 in the currently selected data-base for deletion. If the record number is omitted, the current record will be marked.

While the record is in this state, it will not be accessible for normal processing. It may be recalled by use of the RECALL directive or permanently deleted by use of the PACK directive.

3.2.8 RECALL OF RECORDS

A record may be recalled after it has been marked for deletion as follows:

```
RECALL 5
```

The above directive will unmark record 5 of the currently selected data-bases If the record number is omitted, the current record will be unmarked.

Once unmarked, the record is again available for normal processing.

3.2.9 PACKING DATA-BASES

Packing a data-base will remove all records which have been marked for deletion.

```
PACK
```

This directive will pack and re-index the currently selected data-base.

3.2.10 SORTING DATA-BASES

A data-base may be sorted using any field which has been defined in the data-base.

```
SORT ON MFR
```

This directive will sort the current data-base on a field named MFR in ascending mode. Note that the data is not moved, only the index files are rebuilt.

3.2.10 SORTING DATA-BASES (cont'd)

```
SORT ON MFR ITEM WHOLSALE
```

The directive will perform a nested sort in the order that the variable appear. Up to 8 levels of nested sorting are allowed.

```
SORT OFF
```

This directive will restore the data-base to an unsorted mode. Following this, records will be retrieved in the order of entry.

Note that the SORT is a disk based function and the length of time required is dependant on the physical order of data in the data-base, and on the device being used.

3.2.11 FINDing data in a data-base

A data-base must be sorted for the FIND directive to function.

The FIND looks for a data match using the first field specified when the data-base was sorted.

The FIND directive will position the data-base at the first logical record which contains the data to be found.

```
FIND "GE"
```

Assuming the data-base is sorted and the first field which is sorted is MFR, the data-base will be positioned at the first record which contains the value "GE". If no match is found, EOF is set and will be displayed on the info line and also available for testing in a command file.

3.2.12 SELECTING A DATA-BASE SLOT

There are 5 slots in which a data-base may be used. The desired slot may be selected by the SELECT directive.

SELECT 3 will select slot 3.

There are a group of directives which always function on the current data-base. As an example, MOVE, will move the current data-base position. If you wish to move the data-base in slot 1 and the current selected slot is 3, you must first select slot 1 and then move the data-bases i.e.

3.2.12 SELECTING A DATA-BASE SLOT (cont'd)

```
SELECT 1
MOVE  2
SELECT 3
```

would select the data-base in slot 1, move forward 2 records, and return to the data-base in slot 3.

Data-base record position is not affected by slot selection. In the above example, when you return to slot 3, the data-base is positioned where it was originally.

The commands which operate on the current slot are:

APPEND	BOTTOM	CLOSE
CREATE	DELETE	DISPLAY
EDIT	FIND	MOVE
PACK	PRINT	RECALL
REPLACE	SORT	TOP
USE		

3.2.13 CLOSING A DATA-BASE

The CLOSE directive closes the current data-base. It is this action which causes all related files pertaining to the current data-base to be updated on the device where the data-base resides.

```
CLOSE ALL
```

will close all open data-bases.

If you wish to use a data-base in a slot which already contains a data-base, the data-base currently active in the slot must first be closed.

If the system is turned off, or in some way terminates without closing open data-bases, some index files may not be updated. If this happens the RECOVER directive may be used to recover data

The QUIT directive will automatically close all open data-bases prior to terminating the system.

3.3 LOCAL DATA

LOCAL variables are provided for temporary storage. They also provide a methodology for defining constants which may then be referred to by name.

3.3 LOCAL_DATA (cont'd)

Local variables have a lifetime which is dependent upon where they are defined. If the local is defined from the keyboard or has a "C" in the last field, it will last until the system is re-booted or until a CLEAR LOCAL directive is entered.

If the local is defined in a command procedure, it will last until the procedure which defined it executes a RETURN. All command procedures which are subordinate to the procedure which defined the local, will also have access to the local variable.

Local variables are created implicitly by literals in the REPLACE directive. These locals cease to exist when the replace directive is completed. They must be considered when allocating local space if the default is changed.

3.3.1 CREATING LOCALS

The local attributes are defined in the same order as in the create step and have the same meanings.

i.e. LOCAL PI N 8 5 C

defines a local variable named PI which is of numerical type, occupies a total of 8 characters, and has 5 places to the right of the decimal point. The "C" in the last field specifies that it is a constant and will be retained even if defined as part of a command file.

Note that if the "C" is present, there must not be any temporary variables defined at the time a permanent LOCAL is created. The "C" option is designed to allow constants to be created from a command file and as such, they must be the first local's defined in the command file.

REPLACE PI WITH 3.14159

places the desired value into PI.

Following this process, anytime PI is referred to in a DISPLAY, PRINT, or REPLACE directive, it will assume the value of 3.14159.

3.3.2 LOCAL SPACE ALLOCATION

LOCAL space is allocated from dynamic memory when the first local is defined. The amount of space allocated is that defined in LSPACE of the system status. If this value is changed, it will not take effect if there are currently any local variables defined. Therefore if the local space is changed while local variables exist, the new space will not become effective until a CLEAR LOCAL directive is entered. Note that local space detracts from the amount of dynamic memory available for data-base processing.

3.4 REPLACING_DATA

The REPLACE directive allows data to be replaced into a local or data-base variable.

REPLACE A WITH PI * B

Replaces the variable A with the result of PI times B. In this example all variables are contained in either the local or current data-base.

Data may be freely exchanged between the current records of any data-base in use and any local variables.

The REPLACE directive will solve partially parenthesized strings.

i.e. REPLACE A WITH SQR(B*(C+D))

Any mathematical operator which must be spelled out, must be separated from variables by a parenthesis.

i.e. (A).AND.(B)

Using the STORE data-base as an example, data may be globally replaced by including the ;FOR clause in the directive.

REPLACE LOCATION WITH "BIN #4" ;FOR MFR = "GE"

Will replace the location of all items manufactured by GE to BIN #4.

3.4.1 LITERALS

A literal may be used as a substitute for a variable on the right side of the equation.

i.e. REPLACE A WITH 3.14159 * B

3.4.1 LITERALS (cont'd)

Valid literals are as follows:

Numerical	Any string containing only +-.0123456789.
Character	Any string enclosed by double quotes "HI THERE"
Date	A string in date format "11/21/87"

3.4.2 DATA-BASE VARIABLES

When variables are evaluated and no prefix is specified, LOCAL variables are searched first, and then the currently selected data-base, to locate where the specified variable is contained.

If variables are contained in data-bases which are active but not currently selected, they must be preceded by the slot number. i.e. 2.B would define variable B as being contained in the currently active data-base in select slot 2. The example above could then be expressed as:

```
REPLACE A WITH PI * 2.B
```

The following prefixes are valid:

```
0 = Local variable
1 = Select slot 1
2 = Select slot 2
3 = Select slot 3
4 = Select slot 4
5 = Select slot 5
```

If the target field size is too small to hold the result, an * is placed in the left-most digit of the target variable.

If the mathematical result of the equation has more than 16 zero's to the right or left of the decimal, an * is placed in the right-most digit of the target field.

There are two system variables which are available to the REPLACE directive as follows:

EOF is set to true when the current data-base is positioned at the end of file. This occurs as a data-base is moved to the EOF or when a FIND directive fails.

3.4.2 DATA-BASE VARIABLES (cont'd)

.DATE. allows the system date which is entered initially or modified with the SET directive to be referenced. It is not available for printing or display, but may be used with any of the date functions or referenced as a variable on the right side of the equation.

3.4.3 CHARACTER HANDLING

Character fields may be concatenated together. The TRIM function may be used to trim off trailing blanks. i.e.

```
LOCAL STRING1 C 15
LOCAL STRING2 C 15
REPLACE STRING1 WITH "HI "
REPLACE STRING2 WITH "THERE"
REPLACE STRING1 WITH TRIM(STRING1) ¦ " " ¦ STRING2
```

will result in string1 containing "HI THERE".

If the target field is too small to hold the defined string, excess characters will be truncated.

3.4.4 MATHEMATICAL FUNCTIONS

The following mathematical functions are available:

ARITHMETIC

+	plus
-	minus
*	multiply
/	divide
**	exponentiation
SQR	square-root
LOG	logarithm
ALOG	anti-log
SIN	sine
COS	cosine
TAN	tangent
ATAN	arc-tangent

CHARACTER

¦	concatenation
TRIM	trailing blanks

BOOLEAN

<	LT
>	GT
<>	NE
=	EQ
~	arithmetic

LOGICAL

.NOT.
.AND.
.OR.

DATE

MONTH
DAY
YEAR
.DATE.

3.4.4 MATHEMATICAL FUNCTIONS (cont'd)

Logical functions operate on true/false results generated by boolean operators. i.e. (A=B).OR.(B>C)

Boolean operators return a 0 for false and a 1 for true.

The arithmetic boolean operator (~) returns a -1, 0, 1.

3.5 RETRIEVING DATA

Data may be retrieved from the current records of any data-base which is currently in use. Data contained in local variables may also be retrieved.

It should be noted that it is permissible to have identical field names in data-bases which are in use in different slots i.e. field name "A" may be present in the data-base currently used in slot #2 and also in the data-base currently in use in slot #4. To differentiate between the field names, the slot identifier may precede the field name. i.e. 2.A and 4.A would uniquely identify each variable. Local variables use 0 as an identifier. If the slot identifier is not present, the system will first search the local field-names, and then the field-names of the data-base in the currently selected slot. If the field-name cannot be found an error message will be generated.

3.5.1 DISPLAYING DATA

Data may be displayed on the console by using the DISPLAY directive. Data may be displayed from the current records of any data-base in use or from any local variable. Additionally, data may be displayed sequentially from all records of the data-base currently selected or, from a specified number of records relative to the current position of the currently selected data-base.

i.e. DISPLAY ALL or DISPLAY (n).

If no field-names are specified with the DISPLAY directive, all variables in the currently selected data-base will be displayed. Specifying field-names with the DISPLAY directive, allows selected variables to be displayed.

i.e. DISPLAY ALL ITEM MFR WHOLSALE

3.5.1 DISPLAYING DATA (cont'd)

would display variables ITEM, MFR, AND WHOLSALE from all records of the currently selected data-base.

i.e. DISPLAY ITEM MFR WHOLSALE

would display variables from the current record of the currently selected data-base.

In the above two cases, the local variables would be searched first.

Variables may be displayed globally from the current data-base by including the ;FOR clause in the DISPLAY directive

DISPLAY ITEM WHOLSALE ;FOR MFR = "GE"

Will display variables from each data-base record having GE as a MFR.

When the data is displayed, record numbers will be present to the left of the data. The record numbers may be suppressed via the SET RECNUM directive.

Headers will be displayed above the first line displayed. The heading may be suppressed via the SET HEADING directive.

Spacing between fields is controlled by the SPACES value in the system status. The spacing may be different, with and without headings, if the length of the heading is longer than the data.

3.5.2 PRINTING DATA

The PRINT and DISPLAY directives are similar with the following exceptions:

a. The PRINTER value in the system status defines the printer device attributes.

b. The PAGE value in the system status defines the number of lines per printer page. A top of form will automatically be issued when this number of PRINT directives has been issued.

c. For each line which is printed, a carriage return and line feed are appended to the line.

d. The EJECT directive will position the paper at Top Of Form.

e. The system will attempt to place the entire print statement on one line. If there is too much data for the printer, the result is printer dependent. On our NEC it goes back to the beginning of the line and overprints

3.5.2.1 Printer Control Codes

The printer codes which are supplied with the system represent our best approximation as to your needs. The system has been designed to allow each user to customize the PRINTER data-base to his needs.

There is a directive named PRINTER to facilitate the usage of control codes. The syntax is PRINTER (printer type) where printer type matches the NAME field of a record in the data-base named PRINTER on the system disk.

PRINTER EPSON

The above directive would select the printer control codes for an EPSON type printer. The current slot must be empty for the directive to function.

The available printer names may be displayed by USEing the PRINTER data-base on the system disk and a DISPLAY ALL NAMES directive. The available control codes mnemonics may be examined by a DISPLAY STRUCTURE directive. The data-base must be closed before the PRINTER directive will function.

This data-base contains one record per printer type. When the PRINTER directive is invoked, the contents of the named record will be installed and the field names contained in this data-base will be recognized by the PRINT directive when they are parenthesized. i.e. suppose you have a data-base with variables A, B, and C. Lets further suppose you would like to print these with variable B in bold print. The following print statement will perform this:

PRINT A (B) B (NM) C

Note that B and NM are valid variables in the PRINTER data-base and are the mnemonics for BOLD and NORMAL. If the PRINTER directive has not been executed to install printer control codes, the parenthesized fieldnames will be ignored.

The PRINTER data-base has been designed to accommodate sufficient variable length so that printers with different control code lengths will still function. Nulls (0's) may be used as fill as most printers will ignore these bits. There is spare space reserved in the data-base record for user specific codes to be entered. Care should be taken not to change the size of the record or all data must be reentered. Additional printer types may be added to the PRINTER data-base by appending new records as with any other data-base.

If the PRINTER data-base is re-structured, there are some restrictions which must be noted.

Restrictions are as follows:

NAME must be the first variable and be 10 characters long.

FF, LF, CR must be the next three fields and must be 2 digits each.

The printer data-base must be sorted on NAME.

The X data type defines printer control codes and must always be even in length. The printer control codes are defined in hexadecimal notation in the data-base. When they are printed, they will be transmitted as control codes. If these codes are displayed, they will be ignored. The codes may be intermixed at will with other variables in the PRINT statement.

TI-BASE writes the complete line in 80 character chunks and then writes a record containing a C/R, L/F. If no PRINTER directive has been executed, the standard default values are issued for these codes. If a PRINTER directive has been executed, then whatever was defined in the PRINTER data-base for CR and LF will be used.

Printer codes may not perform exactly the same function on different printers. For instance we found that on our NEC, a L/F only works if you are at the beginning of the carriage. Otherwise it is ignored. On an EPSON, I believe it drops down a line on the printer and retains the horizontal location on the printer.

PRINT (CR) (LF) (LF) (LF) (LF) (LF) (LF) will space 6 lines on our system. Note that the () are not delimiters and a , or blank must delimit these from other variables.

3.5.2.2 Printer Attributes

The printer attributes may be specified as a device or as a file. A file attribute will cause a DV80 file to be created and data will be appended onto the end of this file.

If the attributes are a device, they should include a CR.LF to suppress generation of these codes by the RS-232 card.

3.5.3 SUMMING DATA

Individual numeric variables in a data-base may be summed with the SUM directive.

```
SUM RETAIL
```

Will cause the values of RETAIL in each record of the current data-base to be numerically summed. The summed value will be displayed when the summation is completed.

```
SUM RETAIL ;FOR MFR = "GE"
```

Will selectively sum RETAIL from only those records having "GE" as a value in the MFR field of each record. The complexity of the expression following the ;FOR may have the same complexity as the REPLACE directive.

3.6 COMMAND LANGUAGE

A command language is provided to allow complex functions to be performed repetitively. All directives which are available from the keyboard are also available with the command language. Additionally, WHILE, IF, and CASE directives allow programatic capability.

Basically, the command language is a sequential list of directives which will be performed when the file is executed via the DO directive.

Comment lines may be included by placing an "*" as the first character in a line.

Continuation lines are permitted by including a semicolon as the last non blank character in a line. Up to 256 characters may be concatenated by the use of continuation lines.

Command files may be nested up to 5 levels. i.e. DO (filename) within a command file will execute the named file and return immediately following the "DO".

A RETURN directive causes termination of the active command file when encountered.

Command file execution may be "paused" by depressing the space bar, and resumed by depressing the "S" key.

Command files may be aborted by the "ESCAPE" function key.

3.6.1 COMMAND PROGRAM CREATION

The MODIFY COMMAND directive allows command files to be created.

```
i.e. MODIFY COMMAND (filename)
```

invokes a full screen editor for the creation of command files. The "HELP" function key will provide function key actions while in the editor.

Command file length in this editor is limited by the number of buffers available or the depletion of dynamic memory. Lengthy procedures may be accommodated by nesting command files.

Command files may be created with the Editor/Assembler module. Some limitations exist when using the MODIFY COMMAND editor on these files as follows:

- Lines with more than 40 characters will be truncated.
- Large files may not be capable of being read in by the MODIFY COMMAND editor.

3.6.2 COMMAND LANGUAGE INVOCATION

Command files are invoked via the "DO" directive.

```
i.e. DO TEST
```

would invoke a command file named "TEST".

3.6.3 COMMAND DIRECTIVES

There are several special directives which are only available in a command file.

3.6.3.1 IF

The IF directive allows decision making to be performed. The directive may include an ELSE portion if desired.

```
i.e.  IF A = 2
          (perform true processing)
          ELSE
          (perform false processing)
          ENDIF
```

The expression following the IF directive must resolve to a boolean value.

3.6.3.2 CASE

The CASE directive allows selective processing of directives. CASE directives are included between DOCASE and ENDCASE directives. Each CASE is examined sequentially. The first case which resolves to "true", will be executed. Execution will be continued until a BREAK directive is encountered. Execution will then be discontinued until the ENDCASE directive is encountered.

```
DOCASE
        CASE A = 1
                (processing if A = 1)
                BREAK
        CASE A = 2
                (processing if A = 2)
                BREAK
        CASE A = 3
                (processing if A = 3)
                BREAK
        ENDCASE
```

In the above example, the variable A will be checked sequentially for the values 1, 2, and 3, and the appropriate processing performed.

If no true expression is found, no processing will be performed.

An otherwise function may be simulated by forcing the last case directive to be true. i.e. CASE 1 = 1

3.6.3.3 WHILE

The WHILE directive allows a section of directives to be executed repeatedly. The expression following the WHILE is tested at each iteration to determine if another iteration is desired.

The normal usage of this directive, is to provide an expression which is modified within the section of directives to be executed. In this way, a set of directives may be executed until a specific event, such as end of file or a data condition, occura. When the defined condition is "false", processing is continued at the first statement following the ENDWHILE.

```
WHILE .NOT. (EOF)
    (directive processing)
    MOVE
    ENDWHILE
```

The above example assumes that there is a data-base in use. The directive processing might include some type of manipulation of data within the current data-base record.

Note that the MOVE directive included in the WHILE loop moves the data-base through sequential records until an end of file is reached.

The system variable, EOF, will test true when the end of file on the currently selected data-base is reached.

3.6.3.4 RETURN

When a RETURN directive is encountered in a command file, the current command file is terminated. Control is returned to the next higher command file. If there is no other command file, control is returned to the keyboard.

3.6.3.5 WAIT

The WAIT directive is not limited to the command language, however it is only useful there.

WAIT 5

This will suspend the system for approximately 5 seconds. The WAIT may be aborted with ESCAPE (F9).

3.6.4 Command Line Trace

The directive TRACE ON/OFF allows a debugging capability within the command line interpreter.

```
TRACE ON
```

Each command line directive will be written to the printer attributes when TRACE is ON. Command lines which are not executed will not be written.

```
TRACE OFF
```

Terminates the command line trace.

These directives may be imbedded within a command file so that a selective portion of a command file may be debugged.

3.7 USER CREATED DISPLAYS

Custom displays may be created from command files in addition to the capability provided by the DISPLAY command. The screen may be cleared and data-base variables and literals may be positioned as desired on the console display. Sections of the display may be scrolled up or down and data may be retrieved in response to prompts.

3.7.1 CLEARING SCREEN

The CLEAR directive will clear the display screen.

3.7.2 SCROLLING

Selected sections of the display may be scrolled by use of the scroll command.

```
SCROLL 5,10
```

will scroll lines 5 through 10 up one line. The data on line 5 will be discarded and line 10 will be blank following the directive.

```
SCROLL 10,5
```

will scroll lines 5 through 10 down one line. The data on line 10 will be discarded and Line 5 will be blank following the directive.

3.7.3 WRITING TO SCREEN

Data-base fields and/or literals may be selectively displayed using the WRITE command.

```
LOCAL A N 5 2
REPLACE A WITH -1.23
WRITE 5,10,"VARIABLE A = ",A
```

The above example would display

```
VARIABLE A = -1.23
```

starting on column 10 of line 5.

3.7.4 READING FROM SCREEN

Data-base fields may be read from the console. These fields may be local variables or fields in any data-base in use.

```
LOCAL A N 5 2
READ 5,10,A
```

The above example will result in the cursor being placed at row 5, column 10, and whatever value entered, placed in local variable A. The variable may then be used or tested as desired. The normal usage of this capability is as a prompt response.

The response to a READ directive may be a numeric literal (+-.0-9), character literal ("character string"), or a fieldname (fieldname)

```
READSTRING 5,10,A
```

The above directive specifically expects a character literal in response and will automatically insert the quotes surrounding the character string.

3.7.5 CHIT-CHAT SUPPRESSION

The echo of command line directives is controlled by the system status value in TALK.

```
SET TALK=OFF
```

will result in suppression of the command line directive echoes.

This permits custom displays to be created without interference from command directive execution.

3.8 DISK MANAGEMENT CAPABILITIES

Included in TI-BASE are disk management capabilities. This allows such things as disk initialization and cataloging to be performed without loading other utilities.

3.8.1 DISK INITIALIZATION

The FORMAT directive allows floppy diskettes to be initialized. A series of prompts will be issued following the directive, to allow the selection of sides, densities, drive number etc. The defaults are for a single sided, single density diskette on drive #2.

Note that this directive is a program segment and requires that the program disk be available when executed. It is not possible to use this directive in a single disk system as the OVRLAY/P file is required during the complete initialization process.

WARNING

> The initialization process uses a large amount of VDP memory during the initialization process. All data-bases in use will automatically be closed prior to the execution of this directive.

3.8.2 DISK CATALOGS

CATALOG (drive). will provide a display of the contents of the diskette in the specified drive.

CATALOG DSK2 will generate a catalog of disk drive 2.

Depressing the space bar during the display will pause the system. Depressing the "S" key will restart the display. The ESCAPE key (F9) will abort the catalog.

3.8.3 COPYING DATA FILES

The COPY directive copies a file.

```
COPY DSK1.TEST/C DSK2.TEST/C
```

Will copy file TEST/C from drive #1 to drive #2.

The copy is a record by record copy process and requires that the source and target devices be present during the entire copy process. A disk to disk copy cannot be performed on a single disk system.

Note that this directive is a program segment and requires that the program disk be available when issued. A prompt will allow the substitution of a data disk in the drive used by the program disk if necessary.

Adding "GO" as the third argument will bypass the substitution prompt if it is not needed.

```
COPY DSK1.TEST/C DSK2.TEST/C GO
```

In this example the program disk is assumed to be available on a drive.

If the device is omitted, the value in system status DATDISK will be used for the device.

3.8.4 DELETING FILES and RECORDS

```
DELETE FILE DSK2.TEST/C
```

will cause the named file to be deleted.

If the device is omitted the value in system status DATDISK will be used for the device.

```
DELETE DATABASE
```

will cause the data-base currently in use and selected to be deleted.

```
DELETE RECORD
```

will mark the current record of the data-base currently in use and selected, for deletion.

```
DELETE RECORD 5
```

will mark record 5 for deletion.

3.8.5 LISTING FILES

LIST DSK1.SETUP/C will list file SETUP/C on disk #1 to the current printer attributes.

The first 80 characters of any file will be listed.

3.9 SYSTEM CAPABILITIES

3.9.1 SNAP

SNAP directive will cause a hardcopy of the current display page to be output to the current printer attributes.

3.9.2 MEMORY

The MEMORY directive causes a display of the current dynamic memory usage. The titles are a little cryptic due to space limitations, but the meanings are as follows:

BFRS is the number of empty buffers available

USED is the number of bytes currently being used.

LEFT is the number of bytes still available

#FRG is the number of fragments the available memory is broken into.

SIZE is the size, in bytes, of the largest fragment.

3.9.3 EXTERNAL FILE INPUT

The CONVERT directive provides the capability of transferring external files into TI-BASE data-bases.

CONVERT (input file) (output data-base) [GO]

i.e. CONVERT TEST NEW GO

Leaving off the GO will allow you to change disks.

The current slot should not be in use and the new data-base should not exist.

The directive will allow you to describe the input file in terms of a structure. It does not allow you to change the order of items in the input file in relation to the data-base. You may define a structure that is smaller than the record size of the input file. In that case data from the input file will be truncated. You may also define a structure which is larger than the input file record size. In this case the data-base will be blank filled where data does not exist.

When the directive completes, the new data-base is not in use. To complete the process you must USE the data-base and RECOVER.

Note that with some manual file copying it should now be possible to add fields to an existing data-base and not lose existing data.

The (input file) must include the complete file name i.e. if you are going to use an old data-base as input, then the input name must include the /D to identify the data file of the data-base. The (output data-base) should not have the suffix attached as it will be automatically appended when the new data-base is created.

SECTION 4

DIRECTIVES

In this section, each directive is listed in alphabetical order. Section 3 and section 5 contain explanations and examples for the more complex directives.

APPEND [BLANK] i.e. APPEND BLANK

Appends a new record onto the current data-base If "BLANK" is omitted, the system enters an interactive mode to allow data to be entered from the keyboard. If "BLANK" is present, a blank record is appended to the data-base and control is returned immediately. This allows data to be "replaced" into the data-base programatically.

BOTTOM

Positions the current data-base at the last logical record.

BREAK

Terminates processing following a "true" CASE directive.

note: may only be used in a command file.

CASE (expression) i.e CASE A > B

If (expression) resolves to a "true" boolean value, the statements following the CASE are executed until a BREAK is encountered.

The DOCASE, CASE, BREAK, and ENDCASE directives permit selective directive executions in command files.

note: may only be used in a command file.

CATALOG (device) i.e. CATALOG DSK2.

Provides a catalog of the files on the specified device.

note: "CATALOG" is a program segment and the program disk is required when the directive is issued.

CHANGE (address)(value) i.e. CHANGE A300 0002

Changes the specified hex address to the specified

hex value. Provides a patch capability.

note: "CHANGE" is a program segment and the program disk is required when the directive is issued.

CLEAR [LOCAL] i.e. CLEAR or CLEAR LOCAL

CLEAR will clear the display screen. CLEAR LOCAL will clear all local variables.

CLOSE [ALL]

Closes the current data-base. If ALL is included, all open data-bases will be closed. All current records are updated for the current data-base on the disk at this time.

COLOR (foreground)(background) i.e. COLOR WHITE LIGHT-BLUE

Changes the display screen colors. The following colors are valid:

transparent	black	medium-green
light-green	dark-blue	light-blue
dark-red	cyan	medium-red
light-red	dark-yellow	light-yellow
dark-green	magenta	gray
white		

note: "COLOR" is a program segment and the program disk is required when the directive is issued.

CONVERT (input file) (output data-base) [GO]

Allows conversion of an external file to a TI-BASE data-base. If "GO" is omitted, a prompt will be issued to allow a disk to be inserted in the device occupied by the system disk. See section 3 for a complete description of this directive.

note "CONVERT" is a program segment and the program disk is required when the directive is issued.

COPY (from file)(to file)[GO] i.e. COPY DSK1.SETUP SETUP GO

Copies from file to file. In the above example, DSK1.SETUP is copied to (DATDISK default device).SETUP. If "GO" is omitted, a prompt will be issued to allow a disk to be inserted in the device occupied by the program disk.

note: "COPY" is a program segment and the program

disk is required when the directive is issued. Disk to disk copy is not possible on a single disk system.

CREATE (filename) i.e. CREATE STOCK

Creates a data-base named "STOCK" on the DATDISK default device. An interactive menu will be invoked which allows the operator to define the structure of the data-base

DELETE (scope)(identifier)

i.e. DELETE DATABASE

Deletes the current data-base

i.e. DELETE RECORD (n)

Marks record (n) in the current data-base for deletion. If (n) is omitted, the current record is marked. Once marked for deletion, the record will not be available for normal processing. The record may be recalled with the RECALL directive until the data-base is packed.

i.e. DELETE FILE (filename)

Deletes the specified file. If no device is specified, the DATDISK default device is used.

note: "DELETE" is a program segment and the program disk is required when the directive is issued.

DISPLAY [ALL] [(fieldname list)] [SCOPE]

or

[STRUCTURE [LOCAL]]

or

[STATUS]

i.e. DISPLAY STRUCTURE

Displays the structure of the current data-base

i.e. DISPLAY STRUCTURE LOCAL

Displays the current LOCAL structure.

i.e. DISPLAY STATUS

Displays the current system status.

i.e. DISPLAY ALL

Displays all fields of all records in the current data-base

i.e. DISPLAY

Displays all fields in the current record of the current data-base

i.e. DISPLAY (n)

Displays all fields of the next (n) records.

i.e. DISPLAY ALL A B C

Displays fields A, B, and C from all records in the current data-base

i.e. DISPLAY A B C ;FOR (A<3).OR.(A>5)

Displays fields A, B, and C from all records in the current data-base in which the value of A is less than 3 or greater than 5. The complexity of the expression following the ;FOR may be that of the REPLACE command.

DO (filename) i.e. DO SETUP

Executes command file SETUP on the DATDISK default file. Command files may be nested 5 deep.

The explicit device may be included if desired.

DOCASE

Initiates a case statement.

note: may only be used in a command file.

EDIT [(n)] i.e. EDIT 5

Edits record (n) of the current data-base. If (n) is omitted, the current record is edited.

EJECT

Causes an eject control code to be written to the printer attributes.

ELSE

The ELSE portion of the IF statement.

note: may only be used in a command file.

ENDCASE

Terminates DOCASE processing.

note: may only be used in a command file.

ENDIF

Terminates IF processing.

note: may only be used in a command file.

ENDWHILE

Terminates WHILE processing.

note: may only be used in a command file.

FIND (expression) i.e. FIND "JONES"

Positions the current sorted data-base to the first record which contains JONES in the sort field. (expression) may be any literal or data-base fieldname. If there is no match between (expression) and the sort field, the data-base will be positioned at EOF.

The FIND directive searches only the first field specified in the SORT directive if the data-base has been sorted in a nested manner.

FORMAT

Initializes diskettes. Prompts will be issued to allow definition of volume name, number of tracks, sides, and density.

note: "FORMAT" is a program segment and the program disk is required when the directive is issued. Initialization of diskettes is not possible on a single disk system.

IF (expression) i.e. IF A=B

Allows selective processing. If the expression is "true", the statements immediately following the IF are executed. If the expression is "false", processing will be suspended until an ELSE or ENDIF is encountered.

note: may only be used in a command file.

LIST (filename)

Will list the contents of the named file to the printer attributes specified in the status display.

Only the first 80 characters of each record will be listed.

note: "LIST" is a program segment and the system disk is required when the directive is issued.

LOCAL (fieldname)(type)(width)(dec)[C] i.e. LOCAL A N 10 2

Defines a local variable. The example creates a variable named "A" which is a numeric type with a total width of 10 characters, with 2 decimal digits. Once defined, the variable may be used interchangeably with data-base fieldnames.

If a "C" is present as the last field, the variable has permanent aspects as opposed to temporary. i.e. the variable will exist until a CLEAR LOCAL or until the system is re-booted

If local variables are defined from the keyboard, they are considered permanent and exist until a CLEAR LOCAL or the system is re-booted

If local variables are defined in a command language, they exist until the command file that created them returns.

MEMORY

Generates a display of the current dynamic memory usage. The titles are a bit cryptic due to memory limitations and the meanings are as follows:

BFRS Is the number of empty buffers available.

USED Is the number of bytes currently being used.

LEFT Is the number of bytes still available.

#FRG Is the number of fragments the available memory is broken into.

SIZE Is the size , in bytes, of the largest fragment.

note: "MEMORY" is a program segment and the system disk is required when the directive is issued.

MODIFY (scope) (filename)

i.e. MODIFY STRUCTURE

Allows the structure of the current data-base to be modified. At the present time, fieldnames may be changed with no impact. If a modification causes the record size to change, existing data will be destroyed.

i.e. MODIFY COMMAND (filename)

Allows a command file to be created or modified. A full screen editor provides editing capability.

MOVE (n) i.e. MOVE 3

Moves the current data-base +- (n) records. If (n) is omitted, the data-base is moved one record forward.

PACK

Permanently removes records which have been marked for deletion from the current data-base

note "PACK" is a program segment and the system disk is required.

PRINT [ALL] [(fieldnames and printer control codes)] [SCOPE]

i.e. PRINT ALL

The print directive is similar to the display directive with the exception that status and structure may not be printed.

Printer control codes may be intermixed with fieldnames as desired. See section 3 for a complete description of this function.

PRINTER (printer name) PRINTER EPSON

Installs the specified printer codes from the data-base named PRINTER on the system disk.

note: "PRINTER" is a program segment and the system disk is required when the directive is issued.

QUIT

Terminates processing in an orderly manner. All open data-bases are closed and TI-BASE exits.

READ (n1)(n2)(expression) i.e. READ 5,10,A

Positions the cursor at line 5, column 10 and accepts variable "A". The variable would normally be a local variable which had been created for this purpose.

The operator response to this may be a numerical literal (-+.0-9), a character literal ("character string"), or a fieldname ((fieldname)).

READSTRING (n1) (n2) (expression)

READSTRING has the same attributes as READ but explicitly expects a character string in response. The necessary quotes are automatically appended.

RECALL (n) i.e. RECALL 5

Unmarks record (n) in the current data-base, which had previously been marked for deletion.

note: "RECALL" is a program segment and the system disk is required when the directive is issued.

RECOVER

Rebuilds the index portion of the structure file of the current data-base Allows recovery of a contaminated data-base in situations where the data file is correct, but the structure file has become contaminated.

note "RECOVER" is a program segment and the program disk is required.

REPLACE (fieldname) WITH (expression) [SCOPE]

REPLACE A WITH SQR(3**2)

In the example, variable "A" is replaced with the result of the expression. Any valid mathematical symbols may be used with any literal, local variable, or data-base fieldname.

To clarify which data-base contains the named variables, the select code is used as a prefix. i.e. 2.A, specifies that variable "A" from the data-base in select slot 2 is to be used. A "0" specifies that the variable is a local.

If the prefix is omitted, local variables are searched first and then the currently selected data-base If the variable is not found an error message will be generated.

The replace string may be up to 255 characters long. If used in a command file the ";" is a continuation character when encountered as the last character in a line.

If the optional SCOPE is included, it is of the form

REPLACE A WITH SQR(3**2) ;FOR (expression).

The expression may be any valid mathematical expression that resolves to a boolean value. i.e. ;FOR (A<3).OR.(A>5).

RETURN

Causes the current command file to return 1 level. If no other command files are active, control is returned to the keyboard.

SCROLL (n1)(n2) i.e. SCROLL 3,10

Scrolls lines 3 thru 10 on the current display screen 1 line up. If the larger line number is specified first, the display will scroll down.

SELECT (n) i.e. SELECT 2

Changes the select slot to 2. Allows multiple data-bases to be active concurrently.

SET (keyword)=(value) i.e. SET DATDISK=DSK2.

Modifies entries in the status table. Valid keywords may be determined by doing a DISPLAY STATUS. Valid keywords are to the left of the = sign.

SNAP

Causes the current display contents to be written to the printer attributes.

note: "SNAP" is a program segment and the system disk must be present when the directive is issued.

SORT (scope)(fieldname list) i.e. SORT OFF

Causes the current data-base to be retrieved in order of entry.

i.e. SORT ON A

Causes the current data-base to be sorted on the field named "A". Sorting is in ascending order. Any fieldname in the current data-base may be used.

Nested sorting will be performed if more than one fieldname is included. Sorting will be performed in the order in which the names appear. Up to 8 levels of nesting are supported.

SUM (fieldname)[SCOPE] i.e. SUM A

Causes the named field to be summed globally for the entire data-base. If SCOPE is present it is of the form ;FOR (expression) and summing will be performed only for the records which solve to a true value of the expression.

i.e. SUM A ;FOR (A<3)

Will sum all values of A which are less than 3.

note: "SUM" is a program segment and the system disk is required when the directive is issued.

TOP

Positions the current data-base to the first logical record.

TRACE [ON/OFF]

TRACE ON

causes each directive executed by the command line interpreter to be also written to the printer attributes.

TRACE OFF

stops tracing.

note: "TRACE" is a program segment and the system disk is required when the directive is issued.

USE (filename) i.e. USE DSK2.ADDRESS

Causes the data-base named ADDRESS on DSK2. to become active in the current select slot. The select slot must have previously been empty.

If the device is omitted, the device specified by DATDISK on the status display will be used.

WAIT (n) i.e. WAIT 5

Suspends processing the specified number of seconds.

WHILE (expression) WHILE .NOT.(EOF)

The statements between the WHILE and ENDWHILE will be executed as long as the expression resolves to "true". In the example, the statements would be repeated until the data-base reaches an end of file. This supposes that there is a "MOVE" included in the statements to advance the record position in the current data-base

WRITE (x)(y)(expression) i.e. WRITE 5,10,"HI THERE"

Writes "HI THERE" at line 5, column 10 of the display screen.

The expression may be any literal or data-base fieldname.

SECTION 5

USEFUL INFORMATION

5.1 DISK ORGANIZATION

The TI-BASE program disk contains several different file types that are identifiable by the slash and letter extension at the end of each program or file name. A list of the different types appears below. It is important to note that any file name you create must be limited to eight characters in length since the two character extension is always added to a file by the TI-BASE program.

File Type	Identifier
Data-base Structure	/S
Data-base Data	/D
Command	/C
Help	/H
Program	/P
Load files	none

Except when copying or deleting files, a file's extension need not be typed in to identify it. For example; if you create a data file called NAMES, it would exist on your data disk as NAMES/D and NAMES/S because TI-BASE knows to add those extensions during the CREATE process. To load (USE) the file, only NAMES would be typed in at the dot prompt. You would NOT enter NAMES/D or NAMES/S. When using the COPY directive, any /S, /D, /C, /H or /P extension MUST be included in the file name that you type in to properly identify the file.

5.2 DYNAMIC MEMORY

TI-BASE uses a dynamic memory area for all current DATA-BASE records, local variables, and command file record editing. The amount of memory available depends on what module was used to load TI-BASE.

The Editor/Assembler and Extended Basic modules provide approximately 2200 bytes of dynamic memory.

Mini-Memory adds 4096 bytes for a total of approximately 6300 bytes of dynamic memory.

The ramifications of this are that the sum of all

5.2 DYNAMIC MEMORY (cont'd)

current records, local storage and command editor use may not exceed the amount of dynamic memory that is available. If problems do arise, some redesign of user defined data-base structures may be required. To determine current memory status you may enter the word MEMORY at the dot prompt and a display of available memory will appear on screen.

5.3 RECOVERING DATA

There may be situations where a data-base structure file has been corrupted by not being closed properly, power failure, software hang-ups (bite your tongue) etc. The most common manifestation of this will be an incorrect number of records shown in the status bar at the base of the screen. If this happens, data may still be recovered if the data-base can be accessed with the USE directive.

```
USE (data-base name)
RECOVER
```

The above sequence will recover as many records as possible from the named data-base as long as the USE directive was successful. Note however, that the data will be recovered in the unsorted mode.

The only side effect is that during the DELETE RECORD and PACK process, there may be old records left at the end of the data-base which will also be recovered. If that is the case you will need to mark the records for deletion again and then re-PACK the file.

5.4 PERFORMANCE CONSIDERATIONS

Since TI-BASE is input/output intensive, meaning that data is continuously read from or written to disk, as opposed to being stored in memory, the type of storage device used will determine the speed with which the system operates and it will determine the permissible size of a data-base.

a. Single-side/single-density disk - worst case.
b. Double-side/double-density disk - is better.
c. Ram-disk or hard disk - best case.

Organizing your data files so that the most used ones have the smallest record sizes will provide better performance. Optimum performance will be realized by having fewer files present on a diskette.

5.5 EXAMPLE COMMAND FILES

The ability to create Command Files for record and data-base processing is what sets TI-BASE apart from the pack, making it unique in the 99/4A world. Command Files are programs that you write within the TI-BASE environment that provide almost unlimited methods of access to the data that you use, in almost any manner that you can conceive.

TI-BASE also allows up to five different data-bases to be open (in different slots) at once and, within the limitations of dynamic memory, supports relational record processing between each of them via Command File processing.

In most cases you will likely use only one data-base at a time, but the power and potential to use up to five files is always within reach with Command Files. Following are some examples of typical directive usage in a Command File environment. Other examples may also be found on the TUTOR disk.

To illustrate the use of directives in Command File programming we will create several sample data-bases. The first will be a publications referencing file named INDEX. The second will be a checkbook management data-base named CHECKS. The structure of each is listed below.

INDEX

SUBJECT	C	28
SOURCE	C	21
TYPE	C	18
DATE	C	5
PAGE	C	3

CHECKS

CHECK#	C	5	
DATE	D	8	
PAIDOUT	N	9	2
DEPOSIT	N	9	2
DESC	C	28	

5.5.1 USING APPEND BLANK

The APPEND BLANK directive is used to add an empty record to the end of the current data-base. An example of its use might be to allow the transfer of records or a portion of a record from one file to another. In this example assume that a second file named DETAIL has already been created. DETAIL has the following file structure;

SUBJECT2	C	28
DETAIL1	C	30
DETAIL2	C	30

5.5.1 USING APPEND BLANK (cont'd)

The purpose of the command file XFER listed below is to copy the data found in the SUBJECT field in the INDEX file and clone it so that it also exists in the same logical record number in the SUBJECT2 field in the DETAIL file.

Note that the use of line numbers in the Command File is done for illustration only. DO NOT use line numbers in ANY command file that you create. Note also, TI-BASE does not require the structured (indented) programming style in order to be able to interpret each line. All Command File programming lines may begin left-justified if you prefer. Lastly, any line that begins with an asterisk (*) denotes a COMMENT within the Command File and thus it is not interpreted as being a part of the executable portion of the Command File.

```
01 * xfer
02 CLEAR
03 LOCAL XFER C 28
04 USE INDEX
05 SORT OFF
06 TOP
07 SELECT 2
08 USE DETAIL
09 SORT OFF
10 TOP
11 * the detail database is assumed to be empty
12 SELECT 1
13 WHILE (.NOT.(EOF))
14    REPLACE XFER WITH SUBJECT
15    SELECT 2
16    APPEND BLANK
17    REPLACE SUBJECT2 WITH XFER
18    SELECT 1
19    MOVE
20 ENDWHILE
21 CLOSE ALL
22 RETURN
```

The result of this command file is that the field SUBJECT in the index data-base has been replicated as SUBJECT2 in the detail data-base. A LOCAL variable has been used in this example for illustration purposes only. It would have been perfectly permissable to replace line 17 with

```
REPLACE SUBJECT2 WITH 1.SUBJECT
```


5.5.2 USING BREAK, CASE, DOCASE and ENDCASE

The CASE directive is used for the evaluation of boolean logic conditions from within a Command File. For example, where a value in the CASE statement is less than, equal to, not equal to or greater than a value in the data file being searched, CASE evaluates for "true". When true exists the Command File statements after CASE are executed until a BREAK directive is found. Thus the CASE statement sets up a condition independent of any data in your data-base that can then be compared to values in your data-base for a true condition.

In the Command File listed below you see an example of how multiple search parameters can be used to scan a file for differing conditions in totally separate fields of data.

```
01 *usecase
02 CLEAR
03 USE INDEX
04 SORT ON TYPE
05 FIND "ARTICLE"
06 WHILE (.NOT.(EOF)) .AND. (TYPE="ARTICLE")
07 DOCASE
08    CASE SUBJECT = "ADVENTURING"
09       DISPLAY SOURCE DATE PAGE
10       BREAK
11    CASE SOURCE = "MICROPENDIUM"
12       DISPLAY SUBJECT DATE PAGE
13       BREAK
14 ENDCASE
15    MOVE
16 ENDWHILE
17 RETURN
```

This Command File loads (USEs) the INDEX file, orders the data on the TYPE field and begins searching for any record that contains the word ARTICLE in the TYPE field. When a "hit" is located, a further examination of the record is made to see if the SUBJECT contains ADVENTURING. If so, the SOURCE, DATE and PAGE data from the record are displayed on screen. If not, the next CASE statement is evaluated for a true condition. If it returns true, meaning that if the SOURCE field data in the record that has already been determined to have ARTICLE in the TYPE field contains MICROPENDIUM, then the SUBJECT, DATE and PAGE data are displayed on screen.

5.5.3 USING THE DELETE DIRECTIVE

The ability to perform global operations within a data-base is another of the many advantages of the Command File programming environment offers. Below is an example of how one might perform a global delete of all records in a data-base.

```
01 *gdelete
02 CLEAR
03 USE INDEX
04 SORT OFF
05 WHILE (.NOT.(EOF))
06    DELETE RECORD
07 ENDWHILE
08 RETURN
```

Because the logic in the DELETE directive does a TOP after each delete-pass, the MOVE directive is not necessary to tell TI-BASE to move to the next record. Instead, each record is marked for deletion that exists at the "top" of the file. As long as the End-Of-File is not reached the DELETE will continue.

Other uses for the DELETE directive include;

DELETE RECORD #-	from the dot prompt to delete a single record.
DELETE DATABASE -	from the dot prompt to delete an entire data base from disk.
DELETE FILE (filename)	from the dot prompt to delete a file from disk.

5.5.4 USING THE FIND and SCROLL DIRECTIVES

With the exception of DISPLAY or PRINT with the FOR clause, FIND is perhaps the most useful directive in the TI-BASE arsenal for locating specific bits of information. In a Command File environment it is used to plant the "seed" for locating any information in an existing data-base. In the next example Command File, FIND is used to locate the first occurrence of the phrase ACCEPT AT in the INDEX file. Once the first occurrence has been located, the WHILE (.NOT.(EOF)) loop continues the search with the .AND. condition ensuring that the original data in the FIND directive is still used for comparison purposes.

5.5.4 USING THE FIND and SCROLL DIRECTIVES (cont'd)

Note that in this Command File we introduce some display amenities to make the data that appears on screen more organized.

```
01 *find
02 CLEAR
03 SET TALK=OFF
04 SET RECNUM=OFF
05 USE INDEX
06 SORT ON SUBJECT
07 FIND "ACCEPT AT"
08 IF (EOF)
09    WRITE 20,2, "No records found..."
10 ELSE
11 WHILE (.NOT.(EOF));
12   .AND.(SUBJECT="ACCEPT AT")
13      DISPLAY SUBJECT
14      DISPLAY SOURCE
15      DISPLAY TYPE
16      DISPLAY DATE PAGE
17      SCROLL 1,17
18    MOVE
19 ENDWHILE
20 ENDIF
21 RETURN
```

This Command File actually illustrates the use of several directives not shown thus far; SET, IF, ELSE, ENDIF, WRITE and SCROLL.

The SET directive is a toggle that turns a default parameters condition ON or OFF. The IF and ENDIF directives provide added conditions to the search as does the ELSE directive. SCROLL makes a neat and very visible display out of an otherwise continuous flow of data on the screen. WRITE prints a message to screen at the location specified by the row/column designators that precede the statement in quotes.

Line 2 in the Command File turns the TALK feature off so that Command File processes are not displayed on screen as they occur. Line 3 eliminates the record number from the left edge of the screen as each record matching the search criteria is found and displayed. Line 9 writes the message in quotes to row 20, beginning at column 2 if no records matching the search criteria are located. Lines 13-16 display the data from any "hits" in a stacked format on screen and then SCROLL in line 17 introduces a blank display space directly underneath the last displayed line of

5.5.4 USING THE FIND and SCROLL DIRECTIVES (cont'd)

the last record located. This provides a visual separator between each match that is displayed.

Note also that we assume HEADING has previously been set OFF to prevent headers with each DISPLAY directive.

5.5.5 USING REPLACE

This directive provides the TI-BASE user with a means of inserting values into defined variables and inserting new information into existing data-base records.

```
01 *replace
02 CLEAR
03 USE INDEX
04 SORT ON SUBJECT
05 FIND "3-D ANIMATION"
06 WHILE (.NOT.(EOF))
07     REPLACE SUBJECT WITH "3D ANIMATION"
08     FIND "3-D ANIMATION"
09 ENDWHILE
10 RETURN
```

In this example, the FIND directive is used twice, once outside of the WHILE...NOT loop to locate the first occurrence of the condition specified, and then repeatedly within the WHILE...NOT loop because the REPLACE directive resets the record pointer after each REPLACE occurs. As you can see from the logic of the program, the phrase "3-D ANIMATION" is REPLACEd with a similar phrase of "3D ANIMATION".

5.5.6 CRUNCHING NUMBERS

Performing mathematical operations within a Command File environment could easily consume an entire chapter all to itself. Listed below is a Command File using the CHECKS data-base mentioned earlier. In this program the LOCALs named PO, DE, BAL and MESSAGE are declared at the beginning of the file. A WHILE...NOT loop is instituted to go through the entire data-base. While that is occurring the values in the PAIDOUT and DEPOSIT fields are being added (totaled) and the running balances stored in PO and DE. The value in BEGBAL in this instance was read in to memory from the SETUP file which was edited to include the following lines;

5.5.6 CRUNCHING NUMBERS (cont'd)

```
01 CLEAR LOCAL
02 LOCAL BEGBAL N 9 2 C
03 REPLACE BEGBAL WITH 1234.56
04 USE CHECKS
05 other command file lines would follow
```

When the End-Of-File is reached the formula in line 12 is used to compute the ending balance in the CHECKS file and then line 13 displays the results on screen.

```
01 *checks
02 LOCAL PO N 9 2
03 LOCAL DE N 9 2
04 LOCAL BAL N 9 2
05 LOCAL MESSAGE C 12
06 REPLACE MESSAGE WITH "BALANCE IS $"
07 WHILE (.NOT.(EOF))
08     REPLACE PO WITH PO+PAIDOUT
09     REPLACE DE WITH DE-DEPOSIT
10     MOVE
11 ENDWHILE
12 REPLACE BAL WITH (BEGBAL+DE)-PO
13 DISPLAY MESSAGE BAL
14 CLOSE
15 RETURN
```

Other mathematical operations such as multiplication, division or any other math function supported by TI-BASE may be performed in a similar manner. Note also, that the SUM directive is available from the dot prompt without having to resort to Command File programming.

5.5.7 USING TRIM and CONCATENATE

TRIM is a TI-BASE directive that is used with the concatenate function to eliminate blank spaces within strings of data. To illustrate, suppose that we have a data-base called NAMES which is structured as follows:

LAST	C	10
MI	C	1
FIRST	C	10

Lets assume that we want to display each entry sequentially with the first name followed by the middle initial and a period, followed by the last name. The following command file will provide this.

```
SET TALK OFF
SET RECNUM OFF
SET HEADING OFF
CLEAR
LOCAL NAME C 20
```

5.5.7 USING TRIM and CONCATENATE (cont'd)

```
USE NAMES
WHILE (.NOT.(EOF))
    REPLACE NAME WITH TRIM(FIRST)¦" "¦MI¦".  "¦LAST
    DISPLAY NAME
    MOVE
ENDWHILE
CLOSE
RETURN
```

Notice that we turned the chit-chat, record numbers and heading off to prevent garbage on the screen. We also assumed that the currently selected slot was empty.

In the above example EOF is a system variable that is testable. It becomes "true" when a MOVE directive causes an End-Of-File. The .NOT. makes the WHILE expression "true" until the file reaches the end of file. Each successive MOVE will access the next record until all of the records in the file have been processed.

In this case we used a LOCAL variable to accumulate the name the way we wanted to display it. The TRIM function drops off the trailing blanks from the first name. We also had to add a blank and a period followed by a blank to display it the way we wanted.

5.6 MULTIPLE DATA BASE PROCESSING

To illustrate the use of multiple data-base usage, we will expand on the previous example. Lets suppose that in addition to having a name in the data-base, that we also have an employee number and a job classification as follows:

```
LAST      C  10
MI        C  1
FIRST     C  10
EMPNUM    N  6
JOBTYPE   N  6
```

Now we add another data-base called ADDRESS structured as follows;

```
EMPNUM    N  6
STREET    C  15
CITY      C  10
STATE     C  2
ZIP       N  5
```

For our last data-base, lets name it JOB and structure it as follows:


```
JOB-TYPE N  6
CLASS    C  80
```

We will also assume that the ADDRESS data-base is sorted on EMPNUM and that the JOB data-base is sorted on JOBTYPE.

The intent of this example is to sequentially process the NAMES data-base, and for each record, position the other two data-bases to related data.

```
CLEAR
SET TALK OFF
SET RECNUM OFF
SET HEADING OFF
CLEAR
LOCAL NAME C 20
SELECT 1
USE NAMES
SELECT 2
USE ADDRESS
SELECT 3
USE JOB
SELECT 1
WHILE .NOT. (EOF)
    SELECT 2
    FIND 1.EMPNUM
    SELECT 3
    FIND 1.JOBTYPE
    SELECT 1
    REPLACE NAME WITH TRIM(FIRST)¦" "¦MI¦".  "¦LAST
    DISPLAY NAME
    DISPLAY 2.STREET,2.CITY,2.STATE,2.ZIP
    DISPLAY 3.CLASS
    MOVE
ENDWHILE
CLOSE ALL
RETURN
```

Notice that we have now added the selecting and using of the two new data-bases. Also within the WHILE loop, we now perform a FIND on these two data-bases. The FIND has the function of positioning the selected data-base to the record which contains the same value as the FIND expression.

We have left out a little error checking here in the interest of clarity. We really should have checked EOF following the FIND as it will be set if the FIND fails. We assumed that a related record existed in each data-base also.

5.6 MULTIPLE_DATA_BASE_PROCESSING (cont'd)

The display functions now display data from all current records. Notice that we specify which slots the data is being displayed from by adding the slot prefix to each field-name.

5.7 NESTED_COMMAND_FILE_USAGE

To illustrate this function, lets take the previous command file and extract the processing within the WHILE loop and put it in another command file as follows;

```
* runnames
SET TALK OFF
SET RECNUM OFF
SET HEADING OFF
CLEAR
LOCAL NAME C 20
SELECT 1
USE NAMES
SELECT 2
USE ADDRESS
SELECT 3
USE JOB
SELECT 1
WHILE .NOT. (EOF)
    DO PROCESS
ENDWHILE
CLOSE ALL
RETURN

* process
REPLACE NAME WITH TRIM(FIRST);" " ; MI ; ". " ; LAST
DISPLAY NAME
SELECT 2
FIND 1.EMPNUM
IF EOF
    REPLACE NAME WITH "no employee data"
    DISPLAY NAME
ELSE
    DISPLAY 2.STREET,2.CITY,2.STATE,2.ZIP
ENDIF
SELECT 3
FIND 1.JOBTYPE
IF EOF
    REPLACE NAME WITH "no job data"
    DISPLAY NAME
ELSE
    DISPLAY 3.CLASS
ENDIF
SELECT 1
MOVE
RETURN
```


5.7 NESTED_COMMAND_FILE_USAGE (cont'd)

This example also illustrates the process of defining a LOCAL in one procedure and accessing it in another.

5.8 SYSTEM_SUSPEND_and_ESCAPE

There are several repetitive operations which may be suspended and restarted.

- o Command file execution
- o Sequential display
- o Sequential print
- o Catalog
- o List

The system is suspended by depressing the space bar until an * appears as the last character on the status bar at the base of the screen.

Restart is accomplished by pressing the "S" key.

The ESCAPE (F9) key will abort operations as follows;

CREATE editor	Return to the dot prompt
MODIFY COMMAND editor	Return to the dot prompt
MODIFY STRUCTURE editor	Return to the dot prompt
APPEND editor	Return to the dot prompt
EDIT editor	Return to the dot prompt
Command file execution	Aborts current cmd file
WAIT state	Clears wait
CATALOG	Return to the dot prompt
LIST	Return to the dot prompt

FILE DESCRIPTIONS

A-1 DATA-BASE FILES

Each data base will consists of two (2) files. The filenames will be identical for each file with the extension differentiating between file types. The files are as follows:

DATA FILE - (filename)/D

The file consists of records, which consist of fields. The fields are as described in the structure file. All data is adjacent and the record size is the sum of the fields. For records of 255 bytes or less, the blocking is performed within the device DSR. For records larger than 255 bytes, TI-BASE performs the blocking internally and records of 255 bytes are used on that file. If a specific record is desired, the block in which it starts and the offset in the block to the start of the record is computed using the record size and block size.

STRUCTURE FILE - (filename)/S

Record 0 of the structure file will contain the descriptions for each field in the data base record. The file organization will be as follows:

Description	Size	Type
Number of fields	1	Binary
Sorted field indicator 0 = no, -1 = sorted	1	Binary
Field 1 type (D,C,N,X)	1	ASCII
Field 1 descriptor	10	ASCII
Field 1 total size	1	Binary
Field 1 # of decimals	1	Binary
.		
.		
.		
Field 17 type (D,C,N,X)	1	ASCII
Field 17 descriptor	10	ASCII
Field 17 total size	1	Binary
Field 17 # of decimals	1	Binary

bytes 222-231 contain the sort field list, which consists of field numbers terminated by a -1 byte.

A-1 DATA-BASE FILES (cont'd)

bytes 232-239 contain the date the file was created.

bytes 240-247 contain the date the file was last changed.

bytes 248-249 contain the record number of the last logical record.

Bytes 250 251 contain the current number of records in the data-base

bytes 252-253 contain the word index of the first logical record in the data-base

Records 1 through 127 are the index records for the file. Each record consists of 2 byte word indexes for each record which point to the next logical record in sequence.

Byte 254 (bytes are numbered 0-254) contains the index record number.

A-2 PROGRAM FILES (filename)/P

The program files are internal fixed records of 255 bytes each. Each record contains a program segment which has been loaded to execute at A000 hex.

A-3 COMMAND FILES (filename)/C

Command files may be either fixed or variable and contain records up to 80 characters in length.

The command file processor will execute files created from various sources such as the Editor/Assemble and TI-BASE MODIFY COMMAND processor.

The MODIFY COMMAND processor in TI-BASE will edit files created by the Editor/Assembler, however, records with lengths of greater than 40 characters will be truncated.

The MODIFY COMMAND processor in TI-BASE creates variable length 80 records with a length of 40.

A-4 HELP FILES (filename)/H

Help files are variable 80 format.

A-5 SYSTEM DISK FILES

The following files are contained on the system diskette.

AID06/H	Help file
AID07A/H	Help file
AID07B/H	Help file
AID07C/H	Help file
AID07D/H	Help file
AID07X/H	Help file
AID08/H	Help file
AID09/H	Help file
AID10/H	Help file
LOAD	Extended Basic load file
LOADTI	TI-BASE second level loader/util
MAIN	TI-BASE fixed file main program
MSGS	TI-BASE fixed file error messages
OVRLAY/P	TI-BASE program segment file
PRINTER/D	Printer data-base data file
PRINTER/S	Printer data-base structure file
SCRN	TI-BASE fixed file VDP setup
SETUP/C	SETUP file executed by TI-BASE
TEST/H	Development test file
TIBASE	Initial load and run loader
TIBASEB	Initial extended BASIC loader
TIBASEP	Initial program image loader

P.O. Box 291610
Pt. Orange, FL 32029