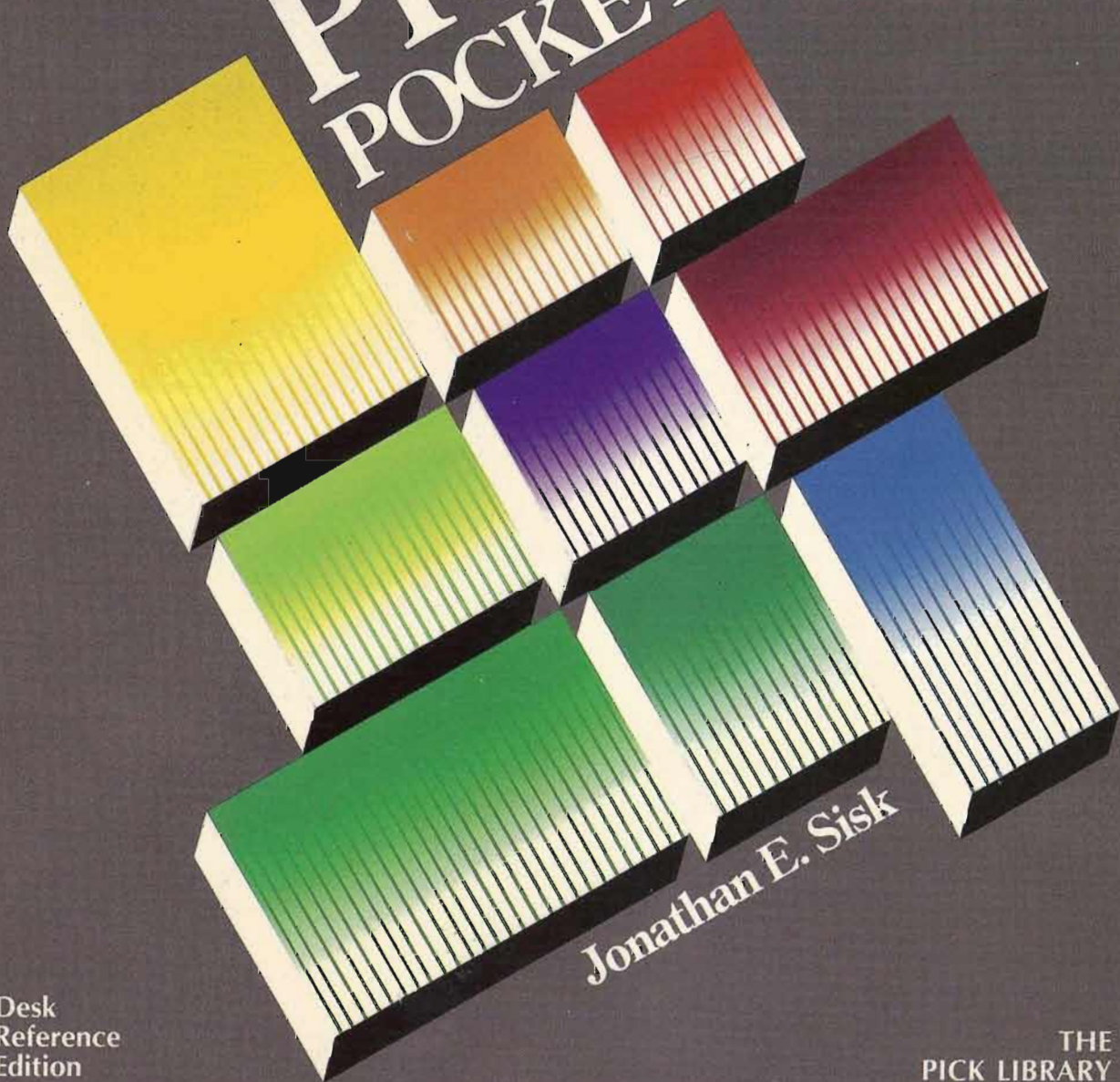


3245

THE PICK[®] POCKET GUIDE



Jonathan E. Sisk

Desk
Reference
Edition

THE
PICK LIBRARY

The
PICK[®]
POCKET GUIDE
JONATHAN E. SISK



TAB Professional and Reference Books

Division of TAB BOOKS Inc.

Blue Ridge Summit, PA

FIRST EDITION
FIRST PRINTING

Copyright © 1982, 1983, 1986, 1988 JES Associates
Copyright © 1989 by TAB BOOKS Inc.
Printed in the United States of America

Reproduction or publication of the content in any manner, without express permission of the publisher, is prohibited. The publisher takes no responsibility for the use of any of the materials or methods described in this book, or for the products thereof.

Library of Congress Cataloging-in-Publication Data

Sisk, Jonathan E.
The PICK pocket guide / by Jonathan E. Sisk. — [4th ed.]
p. cm.
Includes index.
ISBN 0-8306-3245-X
1. PICK (Computer operating system) I. Title.
QA76.76.O63S574 1989
005.4'46—dc19

88-32309
CIP

TAB BOOKS Inc. offers software for sale. For information and a catalog, please contact TAB Software Department, Blue Ridge Summit, PA 17294-0850.

Questions regarding the content of this book
should be addressed to:

Reader Inquiry Branch
TAB BOOKS Inc.
Blue Ridge Summit, PA 17294-0214

Larry Hager: Acquisitions Editor
Pat Mulholland-McCarty: Technical Editor

Notices

The Spectrum Manufacturers Association (SMA) has agreed to allow references to the "standard" Pick features as defined by the SMA standards documentation. Copies of the SMA standards documentation are available for a subscription price of \$25.00 per single copy and includes future updates. Please direct all inquiries for the SMA standards documentation to:

Spectrum Manufacturers Association
10675 Treena Street, Suite 103
San Diego, CA 92131
Phone: (619) 578-3152 FAX: (619) 271-1032

SMA asked us to include the following in the Pocket Guide:

"SMA Standards are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining, with minimum delay, the proper product for his particular need."

"Some material contained herein is designated as proprietary by individual member companies of SMA listed below. Any unauthorized use of such proprietary information is prohibited."

"Copyright ADDS/NCR, Altos Computer, Inc., Automatic Data Processing, Inc., CIE Systems, Inc., Fujitsu Microsystems of America, Inc., General Automation, Inc., IN2 of France, McDonnell Douglas Computer Systems Co., Pick Systems, Inc., Prime Computer, Inc., Pyramid Technology Corp., Scan-Optics, Inc., Sequoia Systems, Inc., Toltec Systems Division of Edge Computer Corp., The Ultimate Corp., Wicat Systems Pacific P/L"

Contents

Preface	ix
Acknowledgments	ix
Conventions and Terms	x
1 Editor	1
Format of TCL Commands Related to the Editor	1
ED	1
EDIT	1
RECOVER-FD	1
About The Terms Used in This Section	2
About SMA Standards	2
About Editor Commands	2
About Delimiters	2
Overview of Editor Commands	2
Editor Error Messages	4

Editor Commands	4	
Using Wild Cards in a Replace	12	
Using Attribute Marks in a Replace	12	
2 PICK/BASIC		17
About The Terms Used in This Section	17	
About Compatibility With the SMA Standard	17	
Format of PICK/BASIC Processing Commands	17	
BASIC	17	
COMPILE	17	
RUN	18	
CATALOG	18	
DECATALOG	19	
About PICK/BASIC Source Code Files	19	
About Object Pointers	19	
Listing Object Code by Date	20	
Reserved Characters	20	
Reserved Words	24	
Precedence and Mathematical Expressions	25	
Notes on Precedence	25	
Logical (Boolean) Expressions	26	
The Search for Truth	26	
Substring Expressions	27	
Array Reference Expressions	27	
Pattern Matching Expression	27	
Relational Operators	28	
Concatenation Operators	28	
Masking Function	28	
Special Uses of Masking	28	
Mask Expression Elements	28	
About Default File Variables	29	
About Statement Labels	30	
About Variable Replacement Statements	30	
PICK/BASIC Statements and Functions	31	
Important Notes About EXECUTE	38	
About PICK/BASIC User Exits	66	
The THEN/ELSE Construct	67	
Single-Line Form	67	
Multi-Line Form	68	
Multiple END statements	69	
3 ACCESS		71
About the Terms Used in This Section	71	
About Compatibility With the SMA Standard	71	
General Format of ACCESS Sentences	72	
Standard Options for ACCESS Commands	72	

Notes on The use of ACCESS Commands	72
Using Quotes in an ACCESS Sentence	72
Logical Operators	73
Special Symbols and Alternate Meanings	74
Logical Connectives AND and OR	74
Throwaway Connectives	75
The USING Connective	75
String Searching	75
Multiple String Searches	76
General format of Attribute-Defining-Items	77
ACCESS Verbs	78
Modifiers and Connectives	84
Correlatives and Conversions	89
About Correlatives	89
About Conversions	89
Format of Attribute-Defining-Items	89
About SMA Standards	91
Operands of the A-correlative	91
Arithmetic Operators	94
Concatenation Operator	94
Arithmetic Functions	94
Relational Operator and Effects	94
Operands of the F-correlative	97
Operators	98
Mathematical Operators and Effects	98
Concatenation Operator	98
Relational Operators and Effects	98
Functional Operators	99
Special Operands	100
User Exits as Correlatives or Conversions	106
User Exits	107
 4 PROC	 109
About SMA Standards	109
About Logon PROCS	109
PROC Linkages	110
PROC Commands	110
 5 RUNOFF	 123
Format of the RUNOFF Command	123
Special Control Characters	124
RUNOFF Commands	124
 6 JET	 133
TCL Commands	133
JET-EDIT	133

	JET-IN	133
	JET-OUT	133
	Key Functions	134
	Moving the Cursor	134
	Special Key Commands	135
	Insert Mode	135
	Replacing Text	135
	Edit Mode	136
	Ruler Edit Commands	136
	Mark Text Commands	136
	Deleting Text	137
	Text Insertion Commands	137
	Overview Options	137
	Backslash (\) Commands	137
7	TCL	147
	Standard TCL Options	147
	About the Terms Used in This Section	148
	Special Cursor Control Functions	148
	TCL Verb Classes	149
	TCL-I Verbs	149
	TCL-II Verbs	149
	ACCESS Verbs	150
	TCL Commands	150
8	The System Debugger	191
	Symbol File Definitions	191
	The SET-SYM Command	191
	Term Conventions and Definitions	192
	The Data Format Specification (DFS)	192
	The Data Reference Specification (DRS)	192
	Address Specifications	192
	Direct Reference	193
	Indirect Reference	193
	The Data Window Specification (DWS)	194
	Offset Specifications	194
	Display Prompts and Special Functions	196
	System Debugger Commands	197
9	PICK/BASIC Debugger	203
	Symbol Definitions	203
	Activating the Debugger	203
	Prompt Character	204
	Operators	204
	Referencing Variables	204

	Zone Output Specification	205
	PICK/BASIC Debugger Commands	205
10	The Spooler	209
	Spooler Commands	209
11	System Error Messages	215
	Error Message Functions	215
	Testing Error Messages	216
	The Sequence of Error Messages	216
	The Standard Pick System Error Messages	216
12	PICK/BASIC Error Messages	241
13	ASCII Table	249
14	Compusheet	257
	Entering CompuSheet +	257
	Spreadsheet Name	257
	Password	258
	Command Menu	259
	Cursor Control	259
	Entering Data, Formulas, and Headings	260
	Editor	260
	Editor Cursor Movement Commands	260
	Editor Commands	261
	CompuSheet + Commands	262
	Formula Entry	268
	Errors in Formulas	268
	Precedence and Precision	268
	Referencing Spreadsheet Cells, Columns and Rows	269
	Literals	269
	Basic Arithmetic Operators	269
	Functions	269
	Math and Trig	271
	String Operations	272
	Reading Information from the Database	272
	Retrieving Data From Other Spreadsheet Cells	273
	Table Lookup	274
	Miscellaneous	274
	Calling PICK/BASIC Subroutines from Formulas	275
	Financial	275
	Referencing Other Formulas	276
	Creating Spreadsheets From TCL	276

SPREAD	276	
SSPREAD	276	
The CS-MENU Process	277	
15 Pick on the IBM AT/XT		281
Changes to Disk Handling	281	
Changes to Streaming Cartridge Tape Handling	281	
Changes to Terminal Assignment	282	
New Extensions to PICK/BASIC	282	
New Extensions to PROC	283	
Changes to the System Debugger	283	
The END Command	283	
The OFF Command	283	
DOS Considerations	283	
About Partitions	283	
The COPYPICK Command	283	
The FDISK Command	284	
Graphics Support	284	
Bootting the Pick System	284	
File Restore Options	285	
Shutting Down the System	286	
Changes to Logon	286	
ABS Frame Usage	286	
Pick XT/AT TCL Commands	287	
Glossary		299
Index		305
About the Author		310

Acknowledgments

I would like to thank John Brandon, Henry Eggers, Terri Hale, Ron Lange, Tim Malnati, Virginia Moreland-Kajikawa, Harvey Rodstein and Ian Sandler for their contributions to this edition of the Pick Pocket Guide.

Preface

This fourth edition of the Pick Pocket Guide includes the complete Spectrum Manufacturers Association (SMA) standards. It also includes all changes to the Pick Operating System up to and including release 2.2. With the printing of this edition, there are now more than fifty-five thousand copies of this book in print. We will do our best to keep it as up to date as possible and will contact you when new versions become available. If you purchased this book through a dealer and would like to be notified of new editions, please contact us and we'll put you on our book replacement list.

I welcome all comments and suggestions. Please direct them to me in care of the publisher.

Conventions and Terms

The following documentation conventions are used consistently throughout *The Pick Pocket Guide*:

<u>Convention</u>	<u>Meaning</u>
{ }	Braces indicate an optional parameter. Typically shown where a parameter must be included in a command or statement. For example: >POVF {(P)}
{ } . . .	Braces followed by three ellipsis points indicate an options parameter which may be repeated.
()	Brackets indicate that the value enclosed is an option. Unless otherwise indicated, options are always the very last argument in a command line, and are enclosed in parentheses.
lowercase italic	User-specified argument or parameter. For example: >MSG <i>acctname text</i>
UPPERCASE	Either means a literal that must be entered exactly as shown or a literal displayed by a process. For example: >CREATE-ACCOUNT<cr>
“	Double quotes are used to indicate a literal parameter.
. .	When two periods appear at the end of a command format line, and the next line of the command also begins with . ., it indicates that the command continues on the next line.

Many terms in the Pick System are unique. Therefore, it is important that you are familiar with the terms & acronyms listed in the glossary at the back of the book. Please refer to the glossary for clarification.

Editor

The following shows the format of Terminal Control Language (TCL) commands related to the Editor:

```
>ED {DICT} filename itemlist* {(options)}
>EDIT {DICT} filename itemlist* {(options)}
```

When the TCL Edit command is invoked, the editor process is activated for entry or update of any file-resident item in the system (i.e. programs, PROCs, data items, etc.). The following is a list of *options* for the TCL commands:

- A Activates the Assembly formatter. Equivalent to the AS editor command.
- M Activates the Macro expansion function. Equivalent to the M editor command.
- P Directs the output to the system printer, via the spooler.
- S Suppresses the display of line numbers, in normal edit mode, or suppresses object code display when the assembly formatter is “on”. Equivalent to the S editor command.
- Z Suppresses the “Top” and EOI” messages.

```
>RECOVER-FD
ITEM NAME: itemname
```

Used to recover an item that was deleted with the editor FD command. It must be executed immediately after the FD command, otherwise the item is lost. If a list of items is active, there is no chance of recovering the item. This does not work on a list deleted through the EDIT-LIST command.

➡➡ About The Terms Used In This Section

Please refer to the glossary located at the back of this book. It contains definitions of the standard terms used in the PICK system, along with the special terms that were created during the production of this guide.

➡➡ About SMA Standards

At the time of this writing, no standards document existed for the Editor. Being one of the oldest components of the system, however, most of these commands work on nearly all versions of Pick.

➡➡ About Editor Commands

All Editor commands may be issued in either upper or lowercase characters and must be followed by a carriage return or linefeed.

➡➡ About Delimiters

Some of the commands described in this chapter might require multiple arguments. These arguments must be separated by a single character called a *delimiter*. The delimiter may be any non-numeric character that does not appear in the string or strings being affected. Commands that require a delimiter include:

DE, FI, FS, L, ME, R

As a documentation standard, throughout this chapter the / character has been used wherever a delimiter is required in one of these commands. This, of course, may be changed as needed when using these commands.

OVERVIEW OF EDITOR COMMANDS

The following table lists the Editor commands & their functions. Each command is detailed in this chapter:

<i>Editor Command</i>	<i>Function/Description</i>
n	Go to line number “n” (Same as Gn)
<cr>	Advance to next line
^	Toggle “wildcard” function
?	Display filename, itemname, line#
A	Again (issue last “Locate” command)
AS	Assembly formatter
B	Bottom
C	Column display
DE	Delete line(s)
EX	Exit current item
EXK	Exit item and return to TCL
F	Flip buffers
FD	Delete current item
FDK	Delete item and return to TCL
FI	File current item
FIK	File item and return to TCL
FIL	File current item as “list”
FIO	File into new item and overwrite
FS	File “save” item and remain in item
FSL	File “save” current item as “list”
FSO	File “save” item and overwrite duplicate
Gn	Go to line number “n”
I	Insert
Ln	List “n” lines
L	Locate
M	Macro expansion
ME	Merge
N	Next
Pn	Prestored command load/activate
PD	Display prestored commands
R	Replace (first occurrence in line)
RU	Replace (all occurrences in line)
S	Suppress line numbers
S?	Display size in bytes
T	Top
TB	Tabstop
X	Oops! Cancels effect of last command
XF	Uh oh! Cancels all changes since last “Flip”
Z	Zone column limits

EDITOR ERROR MESSAGES

COL#?	Appears when a non-numeric character appears in a syntax position that ordinarily requires a number.
CMND?	Short for “command?”. The editor’s equivalent for system error message 3.
EOI n	End of Item occurs at line “n”.
L n	Indicates either the current line number (with the ? command), or the last line affected by an X (oops) command.
NOT ON FILE	Can’t find requested item with the ME (merge) command.
SEQN?	Sequence? Remember that changes <i>must</i> be made from the top down. Issue an F (flip) command and try again.
STRING?	Usually means that the second part of the replace string has been omitted or that the syntax for the merge command is incomplete. Also appears when a non-numeric character appears in a syntax position that ordinarily requires a number.
TOP	Positioned at attribute 0, top of the item.

Wrapup Error Messages

'id' Deleted.	(Error Message item-id = 222)
'id' Exited.	(Error Message item-id = 220)
'id' Filed.	(Error Message item-id = 221)

EDITOR COMMANDS

The rest of this chapter is an examination (in alphabetical order) of the Editor commands and their functions.

<cr>

Advances line pointer to next line and displays line.

^

Toggles the function of the caret “wildcard” search character used in Locate and Replace commands. Initially, this function is “on”.

?

Displays current itemname and line number.

A

Again. Repeats last Locate command.

AS

Assembly format. Toggles assembly format on or off.

B

Bottom. Positions line pointer to end (bottom) of item.

C

Column. Displays terminal columnar positions.

DE#lines{/}

DE#lines/string

DE#lines/string/b-e

Delete. The Delete commands delete a specified number of lines starting at current position (line number). Note that line pointer is not incremented.

A delete command ending in a delimiter displays the lines as they are deleted. If a *string* is specified, only the lines containing the string are deleted. The *b* and *e* parameters respectively indicate the beginning and ending column ranges in which strings are eligible for deletion. The following are examples of the DE Command

DE<cr>

Deletes the current line.

DE10<cr>

Deletes the current and next nine lines.

DE10/<cr>

Deletes and displays the current and next nine lines.

DE10/ABC<cr>

Deletes lines containing the string "ABC" in the current and next nine lines.

DE10/ABC/5-9<cr>

Deletes and displays the lines containing the string "ABC" in column ranges 5 through 9, in the current and next nine lines.

EX

Exits current item without saving any changes. Control returns to the next item in list or to TCL if no more items are in list. See also the EXK command.

EXK

Exits item and kills active item list, if present, without saving any changes. Control returns directly to TCL.

F

Flip. Toggles the current editor buffer, allowing examination of all changes made in item. Positions the line pointer to the beginning (top) of item. Changes to items must be made from the “top down”. This means that changes may not be made to lines above a line in which a change has already been made unless the F command is used. As a general rule, use the F command whenever changes are required above the last line in which a change was made, or upon receiving a “SEQN?” error message.

FD

File Delete. Deletes current item from file. Control returns to next item in list or to TCL if no more items are in list. To recover the item, use the TCL command RECOVER-FD. This command must immediately follow the FD command. It is documented near the beginning of this section.

FDK

File Delete and kill item list. Deletes current item from file and “kills” active item list. Control returns directly to TCL. To recover the item, use the TCL command RECOVER-FD. This command must immediately follow the FDK command. It is documented near the beginning of this section.

FI

FI *itemname*

FI(*filename*

FI(*filename itemname*

Files current item. Control returns to next item in list or to TCL if no more items are in list. Note that a different filename and/or itemname may be specified prior to actually filing the item. See also the FIO command. For example:

```
>ED BP PROGRAM1<cr>
TOP
.FI NEW.PROGRAM1<cr>
'NEW.PROGRAM1' FILED.
```

FIL

Files item in “list” format. Used to convert an item from the standard (data) item format into a list-class format. Caution: be careful with this — use it only for text or PICK/BASIC source code programs. PICK/BASIC cannot read a “list-class” item, yet. Also, it might be necessary to change the “D” pointer of the DATA section of the file to a “DC” before this works properly.

FIO *itemname*

FIO(*filename*

FIO(*filename itemname*

Files current item and overwrites a duplicate item name if it exists already on file. Note that this form only has to be used when specifying an item name other than the one originally requested. See also the FI command. For Example:

```
>ED BP PROGRAM1<cr>
TOP
.FI NEW.PROGRAM1<cr>
CMND?
```

In the above case, the “CMND?” system response does not actually mean that the command is invalid; rather, it means that the item, NEW.PROGRAM1, is already on file. It can be overwritten as follows:

```
.FIO NEW.PROGRAM1<cr>
'NEW.PROGRAM1' FILED.
```

FS

FS *itemname*

FS(*filename*

FS(*filename itemname*

File “Save” current item. Same as FI command, but edit session continues within the current item. Note: When editing large items, it is suggested that an FS command be issued periodically to ensure an update to the file.

A different *filename* and/or *itemname* may be specified prior to actually

saving the item. See also the FSO command. Here is an example of using a different item name:

```
>ED BP MAIN.PROGRAM<cr>
TOP
.FS MAIN.PROGRAM.ORIGINAL<cr>
```

This effectively makes a backup copy of “MAIN.PROGRAM”, calling the new copy “MAIN.PROGRAM.ORIGINAL”.

FSL

Files to be “saved” in “list” format. Used to convert an item from the standard (data) item format into a list-class format. *Caution:* be careful with this—use it only for text or PICK/BASIC source code programs. PICK/BASIC cannot read a “list-class” item, yet. Also, it may be necessary to change the “D” pointer of the DATA section of the file to a “DC” before this works properly. See also the FS command.

FSO *itemname*

FSO(*filename*)

FSO(*filename itemname*)

File “Saves” current item and overwrites a duplicate *itemname* if it exists already on file. Note that this form only has to be used when specifying an itemname other than the one originally requested. See also the FS command. For Example:

```
>ED BP MAIN.PROGRAM<cr>
TOP
.FS MAIN.PROGRAM.ORIGINAL<cr>
CMND?
```

In the above case, the “CMND?” system response does not actually mean that the command was invalid; rather, it means that the item, MAIN.PROGRAM.ORIGINAL, was already on file. It can be overwritten as follows:

```
.FSO MAIN.PROGRAM.ORIGINAL<cr>
```

No indication that the operation took place successfully is displayed. However, have faith; it probably did.

Glinumber

linenumber

Go to the designated *linenumber*. Positions line pointer to specified line number. A number, followed by a <cr>, positions the line pointer to the specified line number, hence, the G command is optional.

I

Insert. Places editor into insert mode for the addition of new attributes to the current item. To return to (editor) command mode, issue a <cr> at the first position of any line. The following is an example of inserting a single line:

```
.I string<cr>
.I<cr>
00n+string<cr>
00n+<cr>
```

To insert a *Null* Line:

```
.I <cr>
```

Note the space following the command. Without it, this will not work.

To insert multiple lines:

Multiple lines may be inserted, if each text string (new line) is delimited with an Attribute Mark (Control {shift} ^). (The shift is only required when the caret is designated as a shifted keyboard character). The following is an example of inserting multiple lines:

```
.I newline1^newline2^newline3<cr>
```

L#lines

L #lines

List displays a specified number of lines. Here's a secret feature: a space may be substituted for the L command: while in the Editor, press the space bar, then enter the number of lines to list, then press <cr>. Legend has it that some lazy programmer left this in the system and didn't mention it to anyone. It seems to have drifted to nearly all of the current Pick implementations.

L/string

L#lines/string

L#lines/string/b

L#lines/string/b-e

Locate. Searches optionally specified number of lines for any occurrence of the specified string. If no *#lines* parameter is entered, the next occurrence is located and displayed. The ending delimiter is required when the string contains trailing blank characters. If the *#lines* parameter is specified, the requested number of lines are searched, and those that match are displayed. The line pointer is incremented by the specified number and may therefore not be positioned at the last line displayed. The *b* and *e* parameters respectively specify the beginning and ending column ranges within which strings are eligible for matching. The caret (^) is used as a wildcard search character within the string parameter for locating or replacing variable strings within strings. It matches any character. The colon (:) is a special delimiter that indicates a column-dependent correspondence. See the following examples of the Locate command:

```
.L/ABC<cr>
```

Locates and displays the next occurrence of the string, "ABC".

```
.L99/ABC<cr>
```

Locates and displays the string, "ABC", in the next 99 lines.

```
.L99/ABC/-5<cr>
```

Locates and displays, in the next 99 lines, the occurrence of the string, "ABC", in column ranges 1 through 5.

```
.L99/A^C<cr>
```

Locates and displays, in the next 99 lines, any line containing a string beginning with the letter "A", followed by any character, and ending with the letter "C".

```
.L:ABC :<cr>
```

Locates the next line containing the string, "ABC", followed by one blank character in the first four positions of the line.

M

Macro. Toggles macro expansion display on or off, for use in assembly listings.

ME{#lines}/{itemname}/{b}

ME{#lines}(filename {itemname()}b)

Merge. Merges a specified number of lines from specified *item*, beginning with line number *b* in *itemname*. Any delimiter, except a left parenthesis, specifies that the parameter to follow is an item name. A left parenthesis specifies a different file name. The following is a list of default specifications that are used if no information is specified by the user:

No *#lines* means to merge 1 line

No *itemname* means the current item name

No *itemname* following *filename* means current item name

No beginning line number (*b*) means to start the merge from the first line.

The following are examples of the MERGE command:

.ME10//<cr>

Merges 10 lines from the current item, beginning with line number 1.

.ME10/ABC/3<cr>

Merges 10 lines from the item called "ABC", beginning with line number 3.

.ME22(APPDOC PROG.DOC)10<cr>

Merges 22 lines from the item called, "PROG.DOC", which resides in the file called "APPDOC", beginning with line number 10.

N{#lines}

Next. Positions the line pointer down a specified number of lines from the current position. An N command, followed by a <cr>, redisplay the current line.

P{rangenumber} command[command]

Prestore (load). Assigns an editor command sequence to the specified *rangenumber* (in the range 0–9). Each prestored command may contain up to 100 bytes. When no range number is specified, zero (0) is assumed. The default for "P0" (or "P") is "L22" (List 22 lines). The Escape key, is used as a command delimiter to separate multiple editor commands.

Note: prestored editor command sequences must be reloaded each time the editor is initiated.

Here are examples of defining prestored commands:

.P1 B[U22[L22<cr>

Stores in prestored command P1 the following command sequence: Bottom of item, Up 22 lines, List 22 lines. Now, each time the command, “P1”, is issued, the last 22 lines of the item are displayed.

.P2 RU9999.ABC.XYZ[F[P2<cr>

Stores in command P2 the following command sequence: replace, in the next 9999 lines, all occurrences of “ABC” with “XYZ”. It then files the item, and repeats the process on the next item, or, if all items have been processed, falls out to TCL.

P{*rangenumber*}

Prestore (activate). Executes a selected prestored editor command sequence. The “P” command, followed by a <cr>, executes the command sequence stored in “P0” (zero), which normally defaults to “L22” (List 22 lines).

PD

Displays the current prestored commands.

R

R{#lines}/oldstr/newstr/){b{-e}}

RU{#lines}/oldstr/newstr/){b{-e}}

Replace. Replaces, in the specified number of lines, the first occurrence of the string referenced in “oldstr” with the string referenced in “newstr”. The “RU” form replaces all occurrences. If the #lines parameter is specified, the line pointer is incremented accordingly. An R command, followed by a <cr>, allows the replacement of an entire line. Entering a <cr> at the first position of the line returns control to the editor command prompt and the line remains unchanged. The “b” and “e” parameters respectively indicate the beginning and ending column ranges in which strings are eligible for replacement. If the “oldstr” parameter is null, then the “newstr” parameter is inserted at the beginning of the line. If the “newstr” parameter is null and the “oldstr” parameter is found, the “oldstr” parameter is deleted from the line.

Note: Multiple replaces may be performed on a line without flipping the buffers with the F command.

➡ Using Wild Cards in a Replace

The caret (^) is used as a wildcard search character within the string parameter, for replacing known and sometimes unknown strings within

strings. It matches any character. Note that the caret is a normal character, not a control character as in the example:

```
.R/^^^//<cr>.
```

See also the ^ command.

➡➡➡ Using Attribute Marks in a Replace

When the control key is pressed and followed by the caret key (and the shift key, if required) the result represents an attribute mark in the editor. Used with the replace command, it has the effect of terminating the line as in the example:

```
.R/ABC/^<cr>
```

The following are examples of the Replace command

```
.R<cr>
```

Positions cursor at beginning of line for replacement of entire line. A subsequent <cr> leaves the line intact.

```
.R10<cr>
```

Positions cursor at the beginning of each of the next ten lines for entire line replacement.

```
.R/^^^//<cr>
```

Replaces the first 3 characters (wildcards ^ of the current line with null. (i.e. deletes first three characters).

```
.R/ABC/^<cr>
```

Replaces the first occurrence of the string, “ABC”, with an attribute mark (control {shift} caret), terminating the line prior to the string, “ABC”.

```
.R10//ABC<cr>
```

Places the string, “ABC” at the beginning of the next ten lines.

```
.R10/ABC/XYZ<cr>
```

Replaces the *first* occurrence of the string, “ABC”, with the string “XYZ” in the current and the next nine lines.

```
.RU10/ABC/XYZ<cr>
```


Replaces *all* occurrences of the string, “ABC”, with the string, “XYZ”, in the current and the next nine lines.

```
.R10/ABC/XYZ/5-9<cr>
```

Replaces the first occurrence of the string, “ABC”, with the string “XYZ”, in the current and next nine lines, *when* the string is found in column positions 5 through 9.

S

Suppress line numbers. The S command toggles the display of editor line numbers, in normal editor mode, and suppresses display of object code when the assembly formatter is on.

S?

Displays size of item in bytes.

T

Top. Positions the line pointer to the beginning (top) of the item, without flipping buffers. (See also the F command).

TB *tabstop*{ *tabstop* . . . }

Sets one or more tabstops. Each argument is separated by a space. The maximum is 16 tabstops. When the Tab key (or <Ctrl> I) is issued during input, the cursor is positioned to the next defined tabstop. The Tab key has no effect when positioned beyond the last tabstop. For example: (See also the (TCL) TABS command).

```
.TB 3 6 9 12 15 18 21<cr>
```

U#*lines*

Up. Position the line pointer up a specified number of lines from current position. The U command, followed by a <cr>, redisplay the current line.

X

Oops! The X command cancels the effect of the last input, insert, delete, or replace command. The X command will not work after multiple replacements within the same line.

XF

Uh oh! The XF command cancels the effect of all commands executed since the last F (flip) command was issued.

Z

Zb-e

Zone defines the zone display limits to display only the data between the specified beginning (b) and ending (e) column ranges. If both parameters are omitted, the zone function is reset to display the entire line.

PICK/BASIC



Please refer to the term glossary located at the back of this book. It contains definitions of the standard terms used in the PICK system, along with the special terms that were created during the production of this guide.

➤ About Compatibility With the SMA Standard

Some instructions in this chapter might not conform to the SMA standard. Those that are unique to the Pick AT/XT versions are indicated with a compatibility index following the instruction. If no index is provided, the instruction conforms to the SMA standard.

➤ Format Of PICK/BASIC Processing Commands

>BASIC *filename itemlist** {(options)}

>COMPILE¹ *filename itemlist** {(options)}

These two commands activate the PICK/BASIC compiler for translation of source code into object code. The following is a list of *options* for PICK/BASIC processing commands:

¹ May not exist on all Pick systems.

- A Displays object code generated by PICK/BASIC compiler
- C Compiles object code without end of line characters. This makes the PICK/BASIC debugger completely useless for debugging the program. Some implementations have changed this to mean “backward Compatible”
- E Outputs error lines only
- I Shows line numbers for code placed in the program by an INSERT or INCLUDE statement, when used in conjunction with the L option
- L Lists program as it compiles
- M Displays program map of descriptor table and correlates source code lines to generated object code frames
- N Activates NOPAGE function on output to the terminal
- P Directs output to system printer, via spooler
- Q Activates PAGE mode on errors
- S Suppresses symbol table generation
- X Cross-references all variables and places entries in the BSYM file
- Y Provides compilation statistics, including: the start clock time in milliseconds, the size of the code in bytes, the size of the descriptor table, the size of the symbol table in bytes, the number of symbol table entries, and the end clock time in milliseconds

>RUN *filename itemname* {(options)}

Executes a compiled PICK/BASIC program. The following is a list of *options* for the RUN command:

- A Prohibits entry into PICK/BASIC debugger; aborts on error conditions
- D Enters PICK/BASIC debugger prior to execution. This is important when parameters are passed in a CALL statement
- E Enters debugger on any error condition
- I Inhibits variable initialization. Not recommended due to reliability, transportability, and data integrity reasons
- N Activates NOPAGE function, on output to terminal
- P Directs output from PRINT statements to system printer, via spooler
- S Suppresses run-time warning messages
(Very useful for demonstrations, but likely to generate lawsuits when included in live applications.)

>CATALOG *filename itemlist**

Effectively, this creates a “verb” from a compiled PICK/BASIC program, by placing an entry in the current accounts’ Master Dictionary that allows execution of the program by entering the program name at TCL. It is required for any program defined as an external subroutine.² It is not necessary

² Not true on every Pick implementation. Some versions do not require the program to be cataloged if it exists in the same file as the CALL program.

to re-CATALOG a program each time it is compiled. The following is the format for activating A Cataloged PICK/BASIC Program

>*programname* {{*options*}}

The program name may be entered at the TCL prompt character and all options explained under the RUN command are available.

>DECATALOG *filename itemlist**

Deletes object code and catalog pointer if one exists in the MD of the current account.

ABOUT PICK/BASIC SOURCE CODE FILES

The file containing PICK/BASIC source code must be defined as a two-level file, that is, it must have both a Dictionary and DATA section. The (D-pointer) entry in the MD defining the location of the dictionary *must* have a “DC” in attribute one, which defines the file as an object code file.³

ABOUT OBJECT POINTERS

Once a program has been compiled, an entry is placed in the dictionary. This “pointer” item defines the virtual location where the object code is actually stored. The name of the pointer item is the same as that of the source program itself. It has the following format

<i>Pointer</i>	<i>Attribute</i>	<i>Description</i>
1	CC	Object pointer identifier
2	fid	First frame-id of object
3	frames	Number of object frames
4	(null)	
5	hh:mm:ss dd mmm yyyy	Time/date of compile

The Pick system prevents the use of the Editor on these object pointers. Use the LIST-ITEM or SORT-ITEM command on the DICT of the source file to examine them.

³ Again, not true on all implementations. Might need to be tested on your system.

LISTING OBJECT CODE BY DATE

Here's a handy trick to list the object code in order of compile date. It can also be used on the POINTER-FILE to show when lists were saved. Add this item to your MD:

```
PF.DATE
001 A
002 5
003 PF.DATE
004
005
006
007 D2/
008 T11,11]DI
009 R
010 8
```

Note that in line 008, the right bracket is a value mark and entered as <ct1>]. The ACCESS sentence to display the report is:

```
>SORT DICT BP BY PF.DATE PF.DATE<cr>
```

POINTER-FILE may be substituted for BP.

RESERVED CHARACTERS

The following characters have special functions and their use should be avoided in variable definitions

:

The Colon is used as a Concatenation operator and PRINT statement argument delimiter. For example of Concatenation

```
X = "THIS IS"
Y = "CONCATENATION"
Z = X : " " : Y
PRINT Z
```

For example in the PRINT Statement:

```
PRINT "THIS IS" : " " : "CONCATENATION"
```

;

The Semi-Colon Program source line delimiter. Each semi-colon *must* be followed by a legal PICK/BASIC instruction or remark. For example:

```
X = 10 ; Y = 15 ; Z = X * Y ; PRINT Z
```

@

The “At” Sign is used as a Cursor control (intrinsic) function. (See the PRINT statement for arguments). For example:

```
PRINT @(-1) : @(20,0) : "MAIN MENU" :
```

,

The Comma is used as a Tab function. Used in a PRINT statement between arguments to output 18 spaces. For example:

```
PRINT "HELLO" , "THERE"
```

!

The Exclamation Point means logical “OR” statement. Also used to define user-specified remarks. (See “Logical Expressions” in this section). For example of use as an “OR”:

```
IF X < 0 ! X > 10 THEN . . .
```

Which reads: If X is less than 0 *or* X is greater than 10 then . . .

For example of use as a Remark:

```
100 ! Mainline . . . .
```

When used as a remark, the entire line is ignored by the compiler.

&

The Ampersand logical “and” statement. (See “Logical Expressions” in this section). For example:

```
IF X > 0 & X < 10 THEN . . .
```

Which reads: IF X is greater than 0 *and* X is less than 10 then . . .

=

The Equal Sign means logical “equal” operator.

#

The “Pound” Sign means logical “not equal” operator.

—

The Minus Sign is used as a mathematical operator (subtract). Also unary minus.

+

The Plus Sign is used as a mathematical operator (add). Also unary plus.

/

The Forward Slash is used as a mathematical operator (divide).

*

The Asterisk is used as a mathematical operator (multiply). Also used to define user-specified remarks. For example of use in Multiplication:

GROSS.WAGES = HOURS.WORKED * HOURLY.RATE

For example of use as a Remark:

100 * Mainline . . .

When used as a remark, the entire line is ignored by the compiler.

^

The caret is used as a mathematical operator (exponent). See also the PWR function. For example of use in Exponentiation:

X ^ 3

Cubes the value of X.

<

The Left Angle Bracket is used as a logical operator for “less than” and is also used as a dynamic array reference character. For example of use as a “Less Than” operator:

IF X < 0 THEN . . .

For example of use in a dynamic array:

CUSTOMER.ITEM<1> = NAME

>

The Right Angle Bracket is used as a logical operator for “greater than” and also used as a dynamic array reference character. For example of use as a “Greater Than” operator:

```
IF X > Y THEN . . .
```

For example of use in a dynamic array:

```
CUSTOMER.ITEM<1> = NAME
```

,

The Apostrophe or Single Quote is used as a string delimiter. Generally, it does not matter which types of quotes are used on literals (strings) although some instructions, such as HEADING and FOOTING do impose certain restrictions on their use.

“

The Double Quote is used as a string delimiter. Just like single quotes, it does not matter which types of quotes are used on literals (strings) although some instructions, such as HEADING and FOOTING do impose certain restrictions on their use.

\

The Backslash is used as a string delimiter. Generally, it does not matter which types of quotes are used on literals (strings) although some instructions, such as HEADING and FOOTING do impose certain restrictions on their use. See the EXECUTE statement for an example.

[

The Left Bracket is used in text string extraction. (See the “Substring” function in this chapter). For example:

```
IF RESPONSE[1,1] = "Y" THEN PRINTER ON
```

]

The Right Bracket is used in text extraction. See the “Substring” function in this section. For example:

```
IF RESPONSE[1,1] = "Y" THEN PRINTER ON
```

(and)

The Left and Right Parentheses are used in three ways: (1) as a function delimiter, (2) to reference array elements in a dimensioned array, and (3) to change the order of precedence in mathematical calculations. See "Precedence" in this chapter. For example as a function delimiter and dimensioned array reference:

```
PRINT OCONV(CUST.ITEM(TOTAL.SALES),"MR2,$")
```

In this illustration, the OCONV function is the function which acts upon the dimensioned array, CUST.ITEM, referring to the amc derived from the variable, TOTAL.SALES.

For example of changing precedence:

```
AGE = (TODAY - BIRTHDAY) / "365"
```

Without the parentheses in the above calculation, an inaccurate result would occur because multiplication occurs at a higher level of precedence than subtraction.

RESERVED WORDS

The following words and symbols should not, and sometimes *cannot* be used as variable names in PICK/BASIC:

!	COL2	EQ	INPUTERR
*	COM	EQU	INPUTNULL
=	COMMON	EQUATE	INPUTTRAP
@	COS	EXECUTE	INSERT
ABORT	COUNT	EXP	INT
ABS	CRT	EXTRACT	LE
ALPHA	DATA	FIELD	LEN
AND	DATE	FOOTING	LN
ASCII	DCOUNT	FOR	LOCATE
BEGIN	DEBUG	GE	LOCK
BREAK	DELETE	GOSUB	LOCKED
CALL	DIM	GOTO	LOOP
CASE	DIMENSION	GT	LT
CAT	DO	HEADING	MAT
CHAIN	DTX	ICONV	MATCH
CHAR	EBCDIC	IF	MATCHES
CLEAR	ECHO	IN	MATREAD
CLEARFILE	ELSE	INCLUDE	MATREADU
CLOSE	END	INDEX	MATWRITE
COL1	ENTER	INPUT	MATWRITEU

MOD	PROCREAD	REWIND	THEN
NEXT	PROCWRITE	RND	TIME
NE	PROMPT	RQM	TIMEDATE
NOT	PWR	SELECT	TO
NULL	READ	SEQ	TRIM
NUM	READNEXT	SIN	UNLOCK
OCONV	READT	SLEEP	UNTIL
OFF	READU	SPACE	WEOF
ON	READV	SQRT	WHILE
OPEN	READVU	STEP	WRITE
OR	RELEASE	STOP	WRITET
OUT	REM	STR	WRITEU
PAGE	REPEAT	SUBROUTINE	WRITEV
PRECISION	REPLACE	SYSTEM	WRITEVU
PRINT	RETURN	TAN	XTD
PRINTER			

PRECEDENCE AND MATHEMATICAL EXPRESSIONS

<i>Symbol</i>	<i>Precedence</i>	<i>Operation</i>
	0 (highest)	Function evaluation
[]	0	Substring extraction
<>	0	Dynamic array extraction
^	1	Exponentiation
*	2	Multiplication
/	2	Division
+	3	Addition
-	3	Subtraction
	4	Masking
:	5	Concatenation
	6	Relational operators
&	7	Logical AND
!	7	Logical OR

➤➤ Notes On Precedence

For program clarity, it might be useful to use parentheses in complex mathematical operations, rather than relying on the precedence of the operator. Expressions are evaluated in order of precedence unless placed within parentheses. Expressions within the innermost parentheses are evaluated first.

LOGICAL (BOOLEAN) EXPRESSIONS

AND or **&**

OR or **!**

Boolean operators are used to connect conditional expressions. When an AND is present, *both* conditions must evaluate to true. The OR accepts either as true. Here are some examples of these operators:

```
IF X = 1 OR X = "Y" THEN . . .
IF X > 1 AND X < 10 THEN . . .
```

THE SEARCH FOR TRUTH

This short explanation of the meaning of true and false may be important in saving you a lot of time in the future. Traditionally, the result of a logical or boolean expression is considered true if it evaluates to 1 and 0 if false.

In the Pick system, expressions which depend on a result of true or false also will evaluate other values as true or false. This tends to vary somewhat between implementations. In generic Pick, any non-zero integer that is positive or negative, evaluates to true. For example:

```
X = 5
IF X THEN STOP
```

Since X is a non-zero integer, the program takes the THEN branch and stops. In generic Pick, any zero or null value evaluates to false. For instance:

```
Y = ""
IF Y THEN STOP ELSE PRINT "YUP"
```

This prints "YUP" since Y is evaluated as false.

Some implementations additionally evaluate any negative numbers as false, and positive numbers as true. All this means is that you will have to take the "truth test" with your system to determine how it handles true and false. This program will do it:

```
LOOP
  PRINT "Value to test " :
  INPUT VALUE
  UNTIL VALUE = "QUIT" DO
    IF VALUE THEN PRINT "TRUE" ELSE PRINT "FALSE"
  REPEAT
```

Try it with negative numbers, null (<cr>), positive numbers, letters and whatever else you can think of.

SUBSTRING EXPRESSIONS

variable[begexp,lengthexp]

Allows the extraction of a “fixed” string from within another string. The “begexp” (beginning position expression) evaluates to a number indicating the starting position within the string, and the “lengthexp” (length expression) evaluates to a number indicating the number of characters to be retrieved. For example:

```
PRINT "PRINT REPORT ? (Y/N) ":  
INPUT RESPONSE  
IF RESPONSE[1,1] = "Y" THEN PRINTER ON
```

ARRAY REFERENCE EXPRESSIONS

Dimensioned Array Reference. References to dimensioned arrays may take either of the following forms:

arrayvar(attexp)
arrayvar(numexp,numexp)

Dimensioned arrays have a maximum of two dimensions and must be defined first with a DIM or DIMENSION statement. See the DIM, MAT (assignment), MATREAD, and MATWRITE instructions.

Dynamic Array Reference. Reference to dynamic arrays can take any of the following forms:

variable<attexp>
arrayvar<attexp,valexp>
arrayvar<attexp,valexp,subvalexp>

Also see the DEL, DELETE, EXTRACT, INS, INSERT, LOCATE, REPLACE, READ and WRITE statements.

PATTERN MATCHING EXPRESSION

The MATCH or MATCHES pattern-matching relational operators may be invoked in either of the following ways:

expression MATCH matchexpression
expression MATCHES matchexpression

(See also the MATCH or MATCHES statement).

RELATIONAL OPERATORS

The following operators are used to construct conditional expressions.

< or LT Less than
> or GT Greater than
<= or LE Less than or equal to
>= or GE Greater than or equal to
= or EQ Equal to
or NE Not equal to
<> or >< Not equal to

CONCATENATION OPERATORS

The concatenation operators join two string expressions together and can be used in either of the following two ways:

stringexpression **CAT** *stringexpression*
stringexpression : *stringexpression*

MASKING FUNCTION

The masking function performs formatting of output data values using the following general format:

variable mask.expression

➤ Special Uses of Masking

Any ACCESS conversion that begins with an “M” can be used as mask expression, simply by dropping off the letter “M”. For example, the ACCESS conversion “MCU” converts a string to all uppercase characters. This can be used in a PICK/BASIC program as follows:

PRINT NAME “CU”

➤ Mask Expression Elements

Justification . . .
{number of digits after decimal} . .
{scaling factor} . .
{Z (suppresses leading zeros)} . .
{, (inserts commas where appropriate)} . .

{signcode} . .
{ \$ (appends dollar sign)} . .
fill character . .
{length}

The scaling factor defaults to the PRECISION statement of the program. If the length of the mask is omitted, the actual length of the field is used. Here are the other types of masks:

Justifications:

R Right justified
L Left justified

Signcodes:

C Outputs CR after negative values
D Outputs DB after positive values
E Outputs "<" before and ">" after negative values
M Outputs "-" after negative values
N Suppresses leading "-" sign on negative values

Special Fill Characters:

%n Fills with "n" zeros
#n Fills with "n" blanks
*n Fills with "n" asterisks

Any other character may be substituted as a fill character.
Here are some examples of masking:

```
PRINT X "R#12"
```

```
PRINT Y "L#15"
```

```
PRINT Z "R#12Z,E$*12"
```

```
PRINT OCONV( DATE( ), "D2/" ) "R#10"
```

ABOUT DEFAULT FILE VARIABLES

The following group of statements makes use of a feature known as *default file variables*:

CLEARFILE, DELETE, MATREAD, MATREADU, OPEN, READ, READU, RELEASE,
SELECT, WRITE, WRITEV and WRITEVU

Most experts agree that default feature is best when it is not used, but here is a quick discussion of how it works. Usually when a file is opened with an OPEN statement, it is assigned to a file variable for referencing the file later in the program with any of the above statements. This takes the general form:

```
OPEN "CUSTOMERS" TO CUSTOMER.FILE ELSE . . .
```

The operative word here is "TO". It assigns the actual location (represented internally as a base, modulo and separation) to the variable called CUSTOMER.FILE in this case. Later in the program, when an item is read from the file, the file variable appears in the appropriate form of a "read" statement, as in the form:

```
READ CUSTOMER.ITEM FROM CUSTOMER.FILE ELSE . .
```

Here, the operative word is "FROM". With "default" file variables, nothing is explicitly assigned during the OPEN statement, as in the form:

```
OPEN "CUSTOMERS" ELSE . .
```

Hence, any subsequent attempt to read from or write to the file does not require the file variable reference, as illustrated in the form:

```
READ CUSTOMER.ITEM ELSE . . .
```

Naturally, there may be only one default file variable in a PICK/BASIC program. Any subsequent file needed for input or output would have to have an explicitly assigned file variable during the OPEN statement.

Using the default file variable form simply is not advised because it makes programs harder to maintain. By spending the few extra seconds to provide the file variable, you can save countless minutes (maybe hours) later in figuring out exactly what the program is intended to do.

ABOUT STATEMENT LABELS

Any PICK/BASIC statement may include an optional statement label prior to the statement. It should be separated from the statement by one or more spaces. At the time of this writing, generic Pick and SMA support only numeric statement labels. Many implementations, however, now support alphanumeric labels.

ABOUT VARIABLE REPLACEMENT STATEMENTS

There are at least five valid methods of replacing the contents of a variable. They are:

- 1) *variable* = *expression*
- 2) MAT *array.variable* = *array.variable*
- 3) MAT *array.variable* = *expression*
- 4) MAT *array.variable* = MAT *array.variable*
- 5) *dynamic.array.variable*<numexp{,numexp{ . . . ,numexp}}> = *string.expression*

PICK/BASIC STATEMENTS AND FUNCTIONS

The difference between PICK/BASIC statements and functions is relatively simple; If the syntax requires that the instruction be followed by a set of parentheses (optionally containing an argument or arguments), then it is a *function*. For example, the following are functions:

```
ABS(numeric,expression)
RND(numeric,expression)
ICONV(string,expression,conversion.expression)
```

Any instruction that does *not* have to be followed by a set of parentheses is most often a *statement*. For example:

```
PRINT X
INPUT Y
EXECUTE SENTENCE
```

There is at least one peculiar exception to the above rule: one variant of the LOCATE instruction is considered a statement, even though its syntax looks like it should be a function.

ABORT (*errmsg#*)

ABORT (*errmsg#*,*“parameter”*{*“parameter”* . . .})

Terminates program and PROC(s). The parameter(s) are passed to the error message handler and displayed with the message stored in the message number indicated by *errmsg#*. Note that the ABORT statement automatically generates error message “[B1] Run-Time Abort at line n”. See also the STOP statement. Also, see ERRMSG section for available messages.

ABS(*numeric.expression*)

Returns the absolute value of the numeric expression in parentheses.

ALPHA(*expression*)

Returns a one (true), if the *expression* contains only alphabetic characters, or zero (false), if it contains any non-alphabetic characters. See the section called “The Search for Truth” near the beginning of this chapter.

ASCII(*expression*)

Converts an EBCDIC expression to its ASCII equivalent. The inverse of EBCDIC.

BEGIN CASE

Initiates the CASE statement. See the CASE statement. For example:

```
BEGIN CASE
  (comment)
  CASE conditional.expression
    statement(s)
  CASE conditional.expression
    statement(s)
  .
  .
END CASE
```

BREAK OFF

BREAK *expression*

Disables the break key on the CRT. In the expression form, the break key is disabled when the expression evaluates to a false. See the section called “The Search for Truth” near the beginning of this chapter.

BREAK ON

BREAK *expression*

Enables the break key on the CRT. In the expression form, the break key is enabled when the expression evaluates to true. See the section called “The Search for Truth” near the beginning of this chapter.

CALL *cataloged-program-name*

CALL *cataloged-program-name(argument(s))*

Transfers program control, either directly or indirectly, to an external subroutine, passing all arguments in list. Arguments must be separated by commas (,). When passing arguments, the same number must occur in the CALL statement that is declared in the SUBROUTINE statement. There is a maximum of about 200 arguments. See also the SUBROUTINE statement. Examples of a direct CALL are:

```
CALL PROCESS.LINES(ID,ORDER.ITEM(1))
CALL "PROCESS.LINES"(ID,ORDER.ITEM(1))
```

With or without quotes, this example still calls PROCESS.LINES sub-routine.

An example of an indirect CALL is:

```
PROGRAM.VARIABLE = "PROCESS.LINES"  
CALL @PROGRAM.VARIABLE(ID,ORDER.ITEM(1))
```

Avoid using this form if you want to write maintainable, efficient programs.

CASE *conditional.expression*

CASE *conditional.expression* ; *statement* (; *statement* . . .)

Conditional execution function, based on whether the value being tested meets the criteria defined in the conditional expression. For example:

```
BEGIN CASE  
  {comment}  
  CASE condexp  
    statement{s}  
  .  
  .  
  CASE condexp  
    statement{s}  
  .  
  .  
  CASE 1  
    statement{s}  
  .  
  .  
END CASE
```

CASE 1 is often used at the end of a set of CASE statements as the condition to perform if none of the other CASE statements are executed.

CAT or :

Concatenation operator. Used to join strings.

CHAIN *expression*

Transfers control to TCL, which interprets and executes the command defined in the expression. Note that control does not return to the program. See also the EXECUTE statement.

CHAR(*numeric.expression*)

Returns the ASCII character equivalent of a numeric value. The inverse of SEQ. See chapter 13 for the available ASCII codes.

CLEAR

Initializes all variables to zero. Avoid using this. It makes debugging programs nearly impossible.

CLEARFILE (*filevariable*)

Deletes all items in specified file variable, just like the TCL CLEAR-FILE command. The default file variable is cleared if no file variable is specified. (See the section called “About Default File Variables” near the beginning of this chapter).

Be careful with this command. On some implementations, if you clear the dictionary level of a file, it also wipes out the data file’s D-pointers.

COL1()

Returns the numeric column position of the character preceding the substring retrieved in the most recently executed FIELD function. Note that the parentheses following the function are required, and never contain any arguments.

COL2()

Returns the numeric column position of the character following the substring retrieved in the most recently executed FIELD function. Note that the parentheses following the function are required, and never contain any arguments.

COM *variable*{*variable* . . . }

COMMON *variable*{*variable* . . . }

Specifies data elements to share among programs executed by a CHAIN, ENTER, or CALL statement. Also defines the storage order for variables.

Note: When using a CHAIN or ENTER statement to initiate another runtime PICK/BASIC program, the “I” option must be used with the RUN command (at TCL) to prevent the reinitialization of variables in workspace and to keep the COMMON area intact.⁴ Suggestion: Don’t use COMMON in conjunction with CHAIN OR ENTER. The only justification for using COMMON is when it can be INCLUDED or INSERTed into code. Otherwise, it makes maintaining programs a real nuisance.

⁴ Most experts agree that there is actually no way to keep the COMMON space intact when using the I option.

COS(*numeric.expression*)

Calculates the cosine of the angle specified in the numeric expression, and returns the results in degrees.

COUNT(*string.expression1*,*stringexpression2*)

Returns the number of occurrences of *stringexpression2* within *stringexpression1*.

CRT

CRT *expression*

Directs output unconditionally to the terminal. Functions like the PRINT command, but is not affected by the (P) option used with the RUN command at TCL; the PRINTER ON statement; nor the HEADING, FOOTING or PAGE statement.

Note that all of the “@” functions provided with the PRINT statement are allowed. See the PRINT statement for more information.

DATA *expression*{*expression*, . . . }

Holds value(s) for use by subsequent input requests initiated from CHAIN, ENTER, EXECUTE, or INPUT statements. Note that multiple DATA statements are permitted. This is preferable to stacking all the data in one long statement.

DATE()

Returns current system date in internal format. Note that the parentheses following the function are required, and never contain any arguments.

DCOUNT(*string.expression1*,*string.expression2*)

Returns the number of occurrences of the delimiter specified in *string.expression2* within *string.expression1*. The *string.expression2* parameter must be a reserved system delimiter, such as a value mark. The difference between this statement and the COUNT statement is that one is added to the result unless the string referenced in *string.expression1* is null.

DEBUG

Transfers control to PICK/BASIC symbolic debugger. See also the D and E options with the RUN command.

DELETE(arrayvar,attexp)
DELETE(arrayvar,attexp,valexp)
DELETE(arrayvar,attexp,valexp,subvalexp)

Deletes a value at any point in a dynamic array. In this form, the DELETE is always on the right side of an equal (=) sign.

DELETE *idexpression*
DELETE filevar,*idexpression*

Deletes specified item-id expression from optionally-specified file variable. See the section called “About Default File Variables” near the beginning of this section.

DIM *variable(dimension)*(,*variable(dimension)* . . .)
DIM *variable(dimension1,dimension2)*

Defines dimensioned array. In the first form, a single-dimensional vector is established. In the second form, a two-dimensional array or matrix is established. The two-dimensional array has nothing to do with the handling of multi-values or sub-values in conjunction with the MATREAD or MATREADU statements.

DO

Initiator for the REPEAT portion of the LOOP statement. See the LOOP . . WHILE and the LOOP . . UNTIL statements.

DTX(*expression*)

Converts decimal value to its equivalent hexadecimal value. See also the XTD function.

EBCDIC(*expression*)

Converts an ASCII expression to its EBCDIC equivalent. The inverse of ASCII.

ECHO OFF

ECHO *expression*

Disables terminal character echo mode. All characters entered on the keyboard are *not* displayed on the terminal. In the latter form, echo is disabled when the expression evaluates to false. See the section called “The Search for Truth” near the beginning of this chapter.

ECHO ON

ECHO *expression*

Enables terminal character echo mode. All characters entered on the keyboard are displayed on the terminal. In the latter form, echo is enabled when the expression evaluates to true. See the section called “The Search for Truth” near the beginning of this chapter. As of release 2.2, control characters entered while ECHO is off are not echoed.

ELSE

Precedes statements to execute when the IF clause evaluates to false. See the section on the THEN/ELSE Construct and the section called “The Search for Truth” near the beginning of this chapter. For example:

```
IF condexp (THEN statement{s}) {ELSE statement{s}}
```

END

Used as a compiler directive in two ways. As a physical program end (optional). For example:

```
. . . . . statements . . . . .  
. . . . . body of program . . . .  
. . . . . statements . . . . .  
END
```

The trailing END statement is not required on PICK/BASIC source code, but is helpful for program readability.

As a logical end of a multi-line IF. For example:

```
IF expression THEN  
    statement{s}  
.  
.  
END
```

Multi-line THEN/ELSE statements MUST have terminating END statements. See the section on THEN/ELSE Construct at the end of PICK/BASIC.

END CASE

Terminates the CASE statement. Typically:

```
BEGIN CASE  
    CASE condexp  
        statement{s}  
    .  
    .  
END CASE
```


ENTER *cataloged-program-name*

ENTER @*programname.variable*

Transfers control to specified cataloged program. See also the COMMON statement. Must *not* be used from a subroutine.

EQU{ATE} *symbol TO constant*{*symbol TO constant . . .*}

EQU{ATE} *variable TO variable*

Performs constant or variable assignment at compile time. A *symbol* is similar to a variable name, with the exception that it may not be modified by placing it to the left of the equal sign in an assignment expression. Here are some examples:

```
EQU CLEAR.SCREEN TO CHAR(12)
EQU RING.BELL TO CHAR(7)
EQU ATTRIBUTE.MARK TO CHAR(254)
EQU CUSTOMER.NAME.ATTRIBUTE TO 1
EQU INVOICE.NUMBER TO CUSTOMER.ITEM(13)
EQU TITLE TO "PICK POCKET GUIDE"
```

**EXECUTE TCL*expression* {RETURNING *return.variable*} . .
. . {CAPTURING *capture.variable*}**

Issues any valid TCL expression, and continues execution of the program at the next line upon completion. The TCL expression may be any valid TCL command, including ACCESS sentences, verbs, PROCs, or other cataloged PICK/BASIC programs. Any input that may be needed by the EXECUTE statement may be provided with the DATA statement.

The optional RETURNING clause is used to direct any error message numbers generated as a result of the EXECUTE statement into a variable. Each error message item-id is separated by a space. See also the SYSTEM function, parameter 17.

The optional CAPTURING clause is used to direct the output from a process into a variable. Carriage return/linefeed combinations are converted to attribute marks, which allows the variable to be treated like a dynamic array. "Clear screens" are converted to nulls.

➤➤➤ Important Notes about EXECUTE

The SMA standard provides an EXECUTE statement. It is however, not as technically elaborate by definition as the Pick form. It does not provide for the RETURNING or the CAPTURING clauses. Most licensees, however, support these extensions.

Control does *not* return from an EXECUTE that issues an OFF or

LOGTO command. Each level of EXECUTE builds a new process workspace area. As the number of levels increase, so do disk space requirements. The maximum number of workspaces is currently five on most systems. It may be changed with the :TASKINIT verb.

The following are examples of the EXECUTE statement:

To issue a simple TCL command:

```
EXECUTE "SORT STAFF BY NAME NAME TOTAL . .  
      . . SALARY (P)"
```

To pass input from the DATA statement:

```
DATA "(QFILE"  
EXECUTE "COPY NEWAC *"
```

To capture output:

```
EXECUTE "LISTU" CAPTURING OUTPUT
```

To capture error message numbers:

```
EXECUTE "SSELECT INVOICES" RETURNING ERRNUM
```

To use all 3 types of string delimiters:

```
EXECUTE \SORT ORDERS = 'S]' WITH DATE > . .  
      . ."1/1/92"\
```

To process lists:

```
EXECUTE "SSELECT STAFF BY HIRE.DATE BY NAME"  
EXECUTE "SAVE-LIST STAFF.LIST"
```

EXP(*numexp*)

Returns the exponential of the *numexp*, that is, e (2,717) raised to the power of the numeric expression. (e^{numexp}). The inverse of LN.

EXTRACT(*arrayvar*,*attrexp*)

EXTRACT(*arrayvar*,*attrexp*,*valexp*)

EXTRACT(*arrayvar*,*attrexp*,*valexp*,*subvalexp*)

Retrieves a specific element from a dynamic array. The alternate method uses dynamic array reference symbols as in the following three formats:

```
arrayvar<attrexp>  
arrayvar<attrexp,valexp>  
arrayvar<attrexp,valexp,subvalexp>
```

Examples of using dynamic array reference symbols are:

```
PRINT EXTRACT(CUSTOMER.ITEM,1,0,0)
PRINT CUSTOMER.ITEM<1>
```

Both of these examples retrieve the entire first attribute from the dynamic array, CUSTOMER.ITEM.

```
PRINT EXTRACT(CUSTOMER.ITEM(1),1,2,0)
PRINT CUSTOMER.ITEM(1)<1,2>
```

Both of these examples retrieve the second (multi-) value from the first attribute of the dynamic array, CUSTOMER.ITEM.

FIELD(*string.expression*,*search.delimiter*,*numeric.expression*)

Returns a substring from a string expression, by specifying a delimiter and the desired occurrence. The search delimiter may not be a reserved system delimiter (Attribute Mark, Value Mark, or Sub-Value Mark). (See also the COL1() and COL2() functions). Here are some examples:

```
GL.ACCOUNT = "02*4000*11"
COMPANY.CODE = FIELD(GL.ACCOUNT,"*",1)
```

This indicates that everything up to the first asterisk in GL.ACCOUNT should be assigned to the variable "COMPANY.CODE". As a result, afterward, COMPANY.CODE contains the value, "02".

```
ACCT.NUMBER = FIELD(GL.ACCOUNT,"*",2)
```

This indicates that everything from the first to the second asterisk should be stored in the variable, ACCT.NUMBER. Thus, ACCT.NUMBER would contain the value, "4000".

FOOTING *expression*

FOOTING "{*text*} {*options*} . . . }"

Designates text to output at the bottom of each page. Options *must* be enclosed within single quotes, to distinguish them from text. Multiple options may be enclosed in the same set of single quotes. See also the PAGE statement). For example:

```
FOOTING "'LC'Page 'PN' Printed at 'TLC'. .  
. . This end into shredder first.'L'"
```

This issues a blank line prior to the centered footing, which includes the text, "Page", followed by the left-justified page number. On the same centered

line the text, “Printed at”, appears and is subsequently followed by the system time and date. After the time and date, the ‘L’ issues a linefeed and the ‘C’ centers the text “This end into shredder first” on the next line. The final “L” issues an extra blank line at the end of the page or screen.

Here is a list of the standard Pick options for the FOOTING command:

- “ Outputs (one) single quote
- C Centers output line
- D Current date (Format dd mmm yyyy)
- L Carriage return and line feed
- P Current page number, right justified in a field of four blanks
- PN Current page number, left justified
- T Current time and date (Format hh:mm:ss dd mmm yyyy)

The SMA Standard option (in addition to the above) is:

P(n) Current page number, right justified in a field of n blanks. When n is omitted, the default is four.

FOR . . NEXT

Incremental loop function. The syntax is:

```
FOR variable = exp TO exp (STEP exp)
    statement(s)
.
.
NEXT variable
```

The STEP function indicates the value to increment the counter variable on each pass. The default step is one.

FOR . . NEXT . . UNTIL

Conditional incremental loop function. Executes until expression following the UNTIL clause evaluates to true. When used, the UNTIL clause *must* appear in the first line of the FOR . . NEXT loop. See the section called “The Search for Truth” near the beginning of this chapter. The typical format is:

```
FOR variable = exp TO exp (STEP exp) UNTIL exp
    statement(s)
.
.
NEXT variable
```

FOR . . NEXT . . WHILE

Conditional incremental loop function. Executes while the WHILE value clause evaluates to true. When being used, the WHILE statement MUST appear in the first line of the FOR..NEXT loop. See the section called “The Search for Truth” near the beginning of this chapter. The typical format is:

```
FOR variable = exp TO exp (STEP exp) WHILE exp
    statement(s)
.
.
NEXT variable
```

GOSUB *statement.label*

Transfers control to specified local subroutine. See also the RETURN statement.

GOTO *statement.label*

GO TO *statement.label*

Transfers control to specified statement label.

HEADING *expression*

HEADING “{*text*} {*options*} . . .”

Designates text to output at the top of each page. Options *must* be enclosed within single quotes, to distinguish them from text. Multiple options may be enclosed in the same set of single quotes. See also the PAGE statement. For example:

```
HEADING "'LC'Accounts Receivable'LC' Aged Trial Balance'L'"
```

This issues a blank line at the top of the screen or page, then centers the text, “Accounts Receivable”. The next 'L' option issues a linefeed and then centers the text “Aged Trial Balance” on the next line. This is then followed by a blank line. See also the FOOTING statement.

Here is a list of standard Pick options:

- “ Outputs (one) single quote
- C Centers output line
- D Current date (Format dd mmm yyyy)
- L Carriage return and line feed
- P Current page number, right justified in a field of four blanks
- PN Current page number, left justified
- T Current time and date (Format hh:mm:ss dd mmm yyyy)

The SMA standard option (in addition to the above) is:

P(n) Current page number, right justified in a field of n blanks. When n is omitted, the default is four.

ICONV(*expression,conversion.expression*)

Converts expression to internal format, according to conversion code specified in conversion expression. Here are two examples:

```
INPUT CHECK.AMOUNT
CHECK.AMOUNT = ICONV(CHECK.AMOUNT,"MR2")
```

```
INPUT CHECK.DATE
CHECK.DATE = ICONV(CHECK.DATE,"D")
```

The ACCESS Section contains more details on each of the following conversions which are available to the ICONV function:

<i>Code</i>	<i>Function</i>
D	Date conversion
G	Group Extract
L	Length
MC	Mask Characters
ML	Mask, left-justified
MR	Mask, right-justified
MT	Mask Time
MX	Mask Hexadecimal
P	Pattern match
R	Range
T	Text Extraction
T	File Translation
U	User Mode Exit

IF *conditional.expression* THEN *statement{s}* {ELSE *statement{s}*}

Performs conditional execution of one or more statements. (See the section on the THEN/ELSE Construct and the section called “The Search for Truth” near the beginning of this chapter).

IN *variable*

Accepts a single character of input. No prompt character is displayed and no <cr> is required. The character is stored in its ASCII (decimal) format in

the variable. Note that the majority of implementations have implemented this differently than the generic Pick form. The others tend to use the form:

```
INPUT X,0.
```

Compatibility: Pick 2.2 and higher

INCLUDE⁵ {*filename*} *itemname*

Inserts compiled object code from specified *itemname* in the program. When the *filename* is omitted, the current file is assumed. Here are some examples:

```
INCLUDE STANDARD.EQUATES
```

```
INCLUDE FILE.DEFS CUSTOMER.FILE.EQUATES
```

```
INCLUDE FILE.DEFS INVOICE.FILE.EQUATES
```

Compatibility: Pick 2.0 and higher SMA

INDEX(*string.expression*,*substring*,*numeric.expression*)

Searches a string expression for the occurrence of the specified substring indicated in the numeric expression. Returns the starting column position of the matched substring. If the specified substring is not found, then 0 is returned. For example:

```
TEST = INDEX("ABCDEFGH","D",1)
```

This returns the value, 4, to the variable called TEST, because the first occurrence of "D" was found in the fourth position of the string.

INPUT {*@(x,y)*}{:}*variable*{*,length.expression*}{:} {*maskexpression*}

Prompts for the input of data from the terminal, using the default prompt character (?), or the character previously defined by a PROMPT statement.⁶

The "At sign" (@) form of INPUT prompts for the input of data from the terminal, at the coordinates specified in the *x* (column) and *y* (row) parameters. Assigns input to specified variable after performing optional mask function.

The optional colon (:) preceding the input variable indicates to display the "old" value of the variable being input.

⁵ Some implementations use \$INCLUDE rather than INCLUDE

⁶ Many implementations now support an ELSE clause on the INPUT statement. Refer to your system documentation to see if you have this feature.

The *length expression* indicates the length of input to accept. A linefeed is performed when the length of the argument is satisfied. The optional colon (:) character following the “lenexp” inhibits the automatic carriage return and linefeed when completing the input. The cursor remains positioned after the entered value.

If the input data value does not match the optionally-specified mask expression, then the statement re-prompts for input. Mask functions may be any combination of conversions, pattern-matching operators, or text justification. See the section on the Masking Function.

INPUTERR *strexp*

Outputs message, defined in string expression, on status (bottom) line of terminal. Used in conjunction with the INPUT @ statement.

INPUTNULL *character*

Defines null default character as specified character, which is recognized as a null on subsequent INPUT statements. Used in conjunction with INPUT @ statements.

INPUTTRAP *string.expression* GOSUB *statement.label*{*statement.label*} . . .

Tests value of last executed INPUT @ statement. Upon execution, branches to statement label address of local subroutine that corresponds with location in statement label list. For example:

```
INPUTTRAP "ABC" GOSUB 1,2,3
```

INPUTTRAP *string.expression* GOTO *statement.label*{*statement.label*} . . .

Tests value of last executed INPUT @ statement. Upon evaluation, branches to statement label address that corresponds with location in statement label list.

INSERT(*arrayvar*,*attrexp*;*exp*) INSERT(*arrayvar*,*attrexp*,*valexp*;*exp*) INSERT(*arrayvar*,*attrexp*,*valexp*,*subvalexp*,*exp*)

Inserts the element referenced by the *expression* into a specific location in a dynamic array. Note: A - 1 may be specified as the *attrexp*, *valexp*, or *subvalexp*. This causes the *exp* to be inserted as the last element in the respective location. Note that only the last form shown is acknowledged by the SMA standard.

INT(*numexp*)

Returns the integer value of the numeric expression, that is, all the numbers to the left of the decimal point.

LEN(*strex*)

Returns the length of the string in the string expression.

LN(*numexp*)

Returns the natural logarithm (base *e*, which is 2.7183) of the numeric expression. Inverse of EXP.

**LOCATE(*exp*,*arrayvar*{,*attrexp*,{*valex*}*};setvar . .*
*. . {;sequence'}) {THEN statement{s}} ELSE statement{s}***

Locates string expression (*exp*) in array referenced by *arrayvariable*, returning the positional value into *setvar*. If the string expression is not found, the value returned in *setvar* contains the position where the element should be inserted using the INSERT function. See the section on the THEN/ELSE Construct.

Listed below are the possible sequence parameters. (Note that the single quotes around parameters are required)

'AL' Ascending, left-justified
 'AR' Ascending, right-justified
 'DL' Descending, left-justified
 'DR' Descending, right-justified

If no sequence parameter is specified, *setvar* position defaults to end of array element.

The use of the optional *attrexp* and *valex* indicate whether the value returned into *setvar* is a Value Mark Count or a Subvalue Mark Count. If both are omitted, the value returned into *setvar* is an Attribute Mark Count.

Compatibility: Pick 2.0 and lower

The SMA and Pick 2.1 form of LOCATE

LOCATE *exp* IN *arrayvar*{<*attrexp*,*valex*>}, . .
. . *numexp* BY *strex* SETTING *setvar* . .
. . {THEN *statement*{s}} ELSE *statement*{s}

The parameters included in the syntax for the SMA standards are the same as those of the standard Pick form, except for the one called *numexp*. This

numeric expression is *not* optional and it indicates the starting position within the specified item, attribute, or value to begin. In other words, if you were searching attribute 2 for a value and the *numexp* was 3, the LOCATE would begin at the third value of the attribute.

Compatibility: Pick 2.1 and higher SMA

LOCK *lock-number* {THEN/ELSE *statement*{*s*}}

Sets an execution (lock) flag in the range, (0–63). This is used to prevent program re-entrancy, such that only one process may run this program at any given time. If the lock-number expression evaluates to greater than 63, then the result is divided by 64, and the lock-number is equal to the remainder of the equation. See the section on the THEN/ELSE construct.

There are only 48 execution locks in Pick 2.0 and lower releases. Many implementations now provide up to 128 execution locks.

LOOP . . . {UNTIL}

Conditional loop function. Performs specified operations until the UNTIL clause evaluates to true. See the section called “The Search for Truth” near the beginning of this chapter. The typical formats for the loop function are:

```
LOOP {stmt{s}} UNTIL condexp DO {stmt{s}} REPEAT
```

```
LOOP {statement{s}} UNTIL condexp DO  
    statement{s}
```

```
    .  
    .  
REPEAT
```

```
LOOP  
    {statements}  
UNTIL condexp DO  
    {statements}  
    .  
    .  
REPEAT
```

LOOP . . . {WHILE}

Conditional loop function. Performs specified operations as long as the WHILE conditional expression evaluates to true. See the section called “The Search for Truth” near the beginning of this chapter. The following are the general formats for the conditional loop for option:

```
LOOP {stmt{s}} WHILE condexp DO {stmt{s}} REPEAT
```

```
LOOP {statement{s}} WHILE condexp DO  
    statement{s}
```

```
    .
```

```
    .
```

```
REPEAT
```

```
LOOP
```

```
    {statements}
```

```
UNTIL condexp DO
```

```
    {statements}
```

```
    .
```

```
    .
```

```
REPEAT
```

MAT *arrayvar* = *exp*

MAT *arrayvar* = MAT *arrayvar*

Initializes all the locations of a dimensioned array variable to the specified value{s}. When assigning one matrix to another, the array on the right side of the equal sign must be the same size or smaller than the matrix on the left side. See also the DIM, MATREAD and MATWRITE statements. For example:

```
DIM CUSTOMER.ITEM(5)  
MAT CUSTOMER.ITEM = ""
```

MATCH *matchstring*

MATCHES *matchstring*

Pattern matching operator. The *matchstring* may be a composite of literals and/or match operators, appended to length specifications. During processing, the value must be the exact length of the length specifications parameter. Listed below are the matchstring operators:

<i>nA</i>	Accept only “n” alphabetic characters
<i>nN</i>	Accept only “n” numeric characters
<i>nX</i>	Accept “n” wildcards (any character)
<i>literal</i>	Any literal string enclosed in single or double quotes

The *n* parameter specifies the length of the match operator string. A length specification of zero (0) allows any length of the following match operator. When combinations of matchstrings and literals are present, the entire matchstring must be enclosed in double quotes. Shown below are examples of single and multiple pattern matching:

Single Pattern Match:

```
IF X MATCHES "3N" - "2N" - "4N" THEN
  PRINT "VALID SOCIAL SECURITY #"
END ELSE
  PRINT "INVALID SOCIAL SECURITY #"
END
```

Multiple Pattern Matches:

```
IF X MATCHES "10N]1A9N" THEN . . .
```

The multiple match example tests for ten numeric characters or one alphabetic character followed by nine numerics. Note that the character separating the 2 pattern matchstrings is a value mark (]).

```
MATREAD arrayvar FROM {filevar}, idexp . .
. . {LOCKED statement{s}} . .
. . {THEN statement{s}} ELSE statement{s}
```

Reads the specified dimensioned array (item) from the optionally-specified file variable, or the default file variable if none is specified, and stores one attribute per element into the array previously defined with a DIM statement. See also: the MATREADU statement, the section on the THEN/ELSE Construct and the section called "About Default File Variables" near the beginning of this section.

If more attributes are present in the item read than elements in the previously dimensioned array, then the extra attributes are appended to the last array element, with each attribute delimited by an attribute mark. If the number of attributes read is less than the array size, then the remaining array elements are assigned null values.

The ELSE condition is taken when the item is not on file. The THEN condition is taken when the item is read successfully.

As of release 2.2, the LOCKED clause is now supported in the syntax of this statement. This clause occurs before the THEN and/or ELSE clause(s) and specifies the statement(s) to execute if the group (or item in newer versions) is locked when the read is attempted. Earlier versions of generic Pick did not support this feature.

```
MATREADU arrayvar FROM {filevar}, idexp . .
. . {LOCKED statement{s}} . .
. . {THEN statement{s}} ELSE statement{s}
```

Reads the specified dimensioned array (item) from the optionally-specified file variable, or the default file variable if none is specified, and stores one attribute per element into the array previously defined with a DIM statement. The MATREADU statement is identical to the MATREAD statement, except that the *group* is "locked", preventing access to that group by

any other process. Note that most other Pick implementations lock only the *item* during the read, and the *group* during an update. See also: the MATREAD statement, the section on the THEN/ELSE Construct and the section called “About Default File Variables” near the beginning of this section.

If more attributes are present in the item read than elements in the previously dimensioned array, then the extra attributes are appended to the last array element, with each attribute delimited by an attribute mark. If the number of attributes read is less than the array size, then the remaining array elements are assigned null values.

The ELSE condition is taken when the item is not on file. The THEN condition is taken when the item is read successfully.

As of release 2.2, the LOCKED clause is now supported in the syntax of this statement. This clause occurs before the THEN and/or ELSE clause(s) and specifies the statement(s) to execute if the group (or item in newer versions) is locked when the read is attempted. Earlier versions of generic Pick did not support this feature.

MATWRITE *arrayvar* ON (*filevar*),*idexp*

MATWRITEU *arrayvar* ON (*filevar*),*idexp*

Writes dimensioned array (item) into the specified file variable. If the file variable parameter is omitted, the default file variable is used. See the section called “About Default File Variables” near the beginning of this section.

The MATWRITEU statement is identical to the MATWRITE statement, except the group remains locked. See also the RELEASE statement.

MOD(*dividend*,*divisor*)

Calculates the remainder (modulo) of the expression. Same as the REM function.

NEXT *variable*

Terminates a FOR . . NEXT loop. See the FOR . . NEXT statement. The syntax is:

```
FOR variable = exp TO exp (STEP exp)  
    statement{s}  
    .  
    .  
NEXT variable
```

NOT(*logical.expression*)

Reverses the normal effect of a logical expression. Returns true if the (logical) expression evaluates to false. Returns false if the expression evalu-

ates to true. See the section called “The Search for Truth” near the beginning of this chapter. For example:

```
IF NOT(NUM(RESPONSE)) THEN PRINT "NOT NUMERIC"
```

NULL

Compiler directive to perform no operation. Typically used by confused programmers in a single line IF . . THEN . . ELSE construct as in the following example:

```
IF condexp THEN NULL ELSE statement{s} . . .
```

NUM(*exp*)

Evaluates to “true” (1) if the expression results in only numeric values. Evaluates to false (0) if the expression contains any non-numeric characters. See the section called “The Search for Truth” near the beginning of this chapter.

OCONV(*exp,convexp*)

Converts the expression to external format, according to conversion code specified in *convexp*. See ACCESS Section for available conversions. The following are examples of the OCONV command:

```
PRINT OCONV(CHECK.AMOUNT,"MR2,$")
PRINT OCONV(CUST.ITEM(INVOICE.DATE),"D2/")
```

The ACCESS Section contains more details on each of the following conversions which are available to the OCONV function:

<i>Code</i>	<i>Function</i>
D	Date conversion
G	Group Extract
L	Length
MC	Mask Characters. MCT, MCU, MCL, MCP, etc.)
ML	Mask, left-justified
MP	Mask, packed decimal
MR	Mask, right-justified
MT	Mask Time
MX	Mask Hexadecimal
P	Pattern match
R	Range
T	Text Extraction
T	File Translation
U	User Mode Exit

ON *indexexp* GOSUB *stmtlbl*(*stmtlbl* . . .)

Transfers control to a local subroutine statement label according to the positional value of the index expression. The local subroutine must be terminated with a RETURN statement.

If the index expression evaluates to a number less than one, or to a number greater than the number of statement labels, then control passes to the next executable line in the program. For example:

```
ON RESPONSE GOSUB 8000,9000,9500
```

ON *indexexp* GOTO *stmtlbl*(*stmtlbl*,)

Transfers control to a statement label according to the positional value of the index expression. If the index expression evaluates to a number less than one, or to a number greater than the number of statement labels, then control passes to the next executable line in the program. For example:

```
ON RESPONSE GOTO 100,200,300,400,500
```

OPEN ("DICT",)*file.exp* (TO *filevar*) {THEN *statement*{*s*}} ELSE *statement*{*s*}

Opens the specified *filename expression* and associates the file with the indicated *file variable*. If the "DICT" parameter is specified (or implied as an expression), then the dictionary level of the specified file name is opened and assigned to the file variable. If no file variable is specified, then the default file variable is opened and assigned. See also the section on the THEN/ELSE Construct and the section called "About Default File Variables" near the beginning of this section.

OUT *variable*

Outputs a single ASCII character, using the decimal equivalent in the specified variable.

Compatibility: Pick 2.2 and higher

PAGE (*expression*)

Terminates the current page of output, prints the optional FOOTING, positions to top of form, and prints the optional HEADING. The optional expression may be used to change the current value of the page number.

PRECISION *number-of-positions*

Defines number of decimal places to carry in results of mathematical expressions. As of release 2.2, the number-of-positions parameter must be in the range 0 to 6. The default precision is four (4). Some older Pick implementations only support precision in the range, 0 through 4.

PRINT ON *printfilenumber print.expression*

Directs printer output to one of 255 print reports in the range (0–254).

PRINT {*@(col,row):*} {*exp,exp . . .*} {*maskexp*}

Outputs literal or expression(s) to the current output device. If the @ symbol is specified, the output is directed to the specified column and row coordinates. If nothing follows the PRINT statement, a blank line is output.

The comma (,) between expressions specifies for the cursor to “tab right” 18 spaces prior to printing the next expression. The colon (:) between expressions and parameters specifies that expressions are concatenated. Note that a colon must appear between the *@(col,row)* parameter and the following expression.

The mask expression may be any legal mask expression. See the section on the Masking function. See also the CRT, PRINTER ON and EXECUTE statements. The following is a list of special control functions of the PRINT Statement:

<u>Function</u>	<u>Description</u>
<i>@(-1)</i>	Clears the screen and positions the cursor to home position (0,0)
<i>@(-2)</i>	Positions the cursor to the home position (0,0)
<i>@(-3)</i>	Clears to the end of screen from the current cursor position
<i>@(-4)</i>	Clears to the end of the line from the current cursor position

Compatibility of above Functions⁷: Pick SMA
The following functions are compatible only with generic Pick systems:

<i>@(-5)</i>	Enables the blink function
<i>@(-6)</i>	Disables the blink function
<i>@(-7)</i>	Enables the protect function
<i>@(-8)</i>	Disables the protect function
<i>@(-9)</i>	Moves the cursor position back one position
<i>@(-10)</i>	Moves the cursor up one line

continued

⁷ The area of the “At minus” functions is perhaps the most non-standardized of all features in this language between various Pick implementations. Rumor has it that at least one Pick licensee now has “At minus” functions through (-732).

If problems are experienced when attempting the above functions, make sure that the *termtype* parameter is correctly set for the type of terminal being used. See the TERM command in the TCL section. The following additional control functions have been added to the PRINT @ statement on the Pick XT and AT releases:

@(- 11)	Activate screen protect
@(- 12)	Deactivate screen protect
@(- 13)	Activate reverse video
@(- 14)	Deactivate reverse video
@(- 15)	Activate underlining
@(- 16)	Deactivate underlining
@(- 17)	Activate slave printer
@(- 18)	Deactivate slave printer
@(- 19)	Move cursor right
@(- 20)	Move cursor down
@(- 21)	Activate graphic character set
@(- 22)	Deactivate graphic character set
@(- 23)	Keyboard lock
@(- 24)	Keyboard unlock
@(- 25)	Control character enable
@(- 26)	Control character disable
@(- 27)	Write status line
@(- 28)	Clear status line
@(- 29)	Initialize terminal mode
@(- 30)	Download function keys
@(- 31)	Non-embedded stand-out on
@(- 32)	Non-embedded stand-out off

The following control functions may only be used on Port 0 (zero), equipped with a color graphics display on the Pick XT or AT:

Background Color Controls:

@(- 33)	White	@(- 37)	Cyan
@(- 34)	Brown	@(- 38)	Green
@(- 35)	Magenta	@(- 39)	Blue
@(- 36)	Red	@(- 40)	Black

Foreground Color Controls (Full Intensity):

@(- 41)	White	@(- 45)	Cyan
@(- 42)	Brown	@(- 46)	Green
@(- 43)	Magenta	@(- 47)	Blue
@(- 44)	Red	@(- 48)	Black

Foreground Color Controls (Half Intensity):

@(- 57)	White	@(- 61)	Cyan
@(- 58)	Brown	@(- 62)	Green
@(- 59)	Magenta	@(- 63)	Blue
@(- 60)	Red	@(- 64)	Black

Additional Control Functions:

- @(- 89) Activate monochrome display unit in 80 by 25 black and white mode
- @(- 93) Activate color graphics monitor in 80 by 25 color mode
- @(- 94) Activate color graphics monitor in 80 by 25 black and white mode
- @(- 99) Returns a 1 if the terminal uses embedded attributes or 0 (zero) if it does not. Returns null if not defined through DEFINE-TERMINAL
- @(- 100) Half-intensity (except on memory mapped monitor)
- @(- 101) Full-intensity (except on memory mapped monitor)

PRINTER CLOSE

Indicates that the current output is now complete, closes the spooler entry, and releases control to the spooler.

PRINTER OFF

Directs output from subsequent PRINT statements to the terminal. See also the CRT statement.

PRINTER ON

Directs output from subsequent PRINT statements to the system printer, via the spooler. See also the CRT statement.

PROCREAD *variable* {THEN *statement*{*s*}} ELSE *statement*{*s*}

Reads the calling PROC's primary input buffer and assigns its contents to the specified variable. When a successful read takes place, the variable is treated as a string of characters delimited by spaces. The FIELD function may be used to parse the variable. The INDEX function may be used to determine the number of entries (# of spaces + 1) in the (buffer) variable.

The ELSE condition is taken when the program has not been executed from a PROC. Note that either a THEN or an ELSE construct must be present. See the section on the THEN/ELSE construct.

PROCWRITE *variable* {ELSE *statement*{*s*}}

Writes the specified string expression variable to the calling PROC's primary input buffer. Note that in order for PROC to be able to deal with the information in the variable, each space in the variable defines a new parameter in the PROC's buffer. The ELSE statement{s} is (are) taken when the program has *not* been run from a PROC.

PROMPT *exp*

Specifies the single character to display during subsequent INPUT statements that prompt for terminal input. The default prompt character is a question mark (?), which is used when the PROMPT statement is NOT used. A null prompt character may be defined with the statement, PROMPT "". Here are some examples:

```
PROMPT CHAR(0)
PROMPT ""
PROMPT ":",
```

PWR(*exp1*,*exp2*)

Raises value contained in *exp1* to the power of the value of *exp2*. See also the ^ character in the section called "Reserved Characters".

READNEXT *id-variable*{*posvar*} {FROM *selectvar*} . .
. . {THEN *statement*{*s*}} ELSE *statement*{*s*}

Reads the next item-id from the active list and assigns it to the specified variable. The list must have been retrieved by a SELECT, SSELECT, GET-LIST, or QSELECT command executed immediately prior to starting execution of the PICK/BASIC program that contains the READNEXT statement, or by a preceding SELECT or EXECUTE statement within the program. See also the (PICK/BASIC) SELECT statement and the section on the THEN/ELSE Construct.

The position variable (*posvar*) parameter indicates the position of the multivalue within an attribute, specified in an exploded sort, executed prior to the execution of the program. This allows multivalues to be retrieved in exploded sort sequence. The ELSE condition is executed if no selection was performed or when there are no more items in the list.

READ *arrayvar* FROM {*filevar*,} *idexp* . .
. . {LOCKED *statement*{*s*}} . .
. . {THEN *statement*{*s*}} ELSE *statement*{*s*}

Reads item from the optionally-specified file variable and assigns each attribute to an array element in the dynamic array designated in *arrayvar*. See also the READU statement and the section on the THEN/ELSE Construct.

If the file variable parameter is not specified, the default file variable is used. See the section called "About Default File Variables" near the beginning of this section.

The ELSE condition is taken when the item is not on file. The THEN condition is taken when the item is read successfully.

As of release 2.2, the LOCKED clause is now supported in the syntax of

this statement. This clause occurs before the THEN and/or ELSE clause(s) and specifies the statement(s) to execute if the group (or item in newer versions) is locked when the read is attempted. Earlier versions of generic Pick did not support this feature.

READT *variable* {THEN *stmt*{s}} ELSE *statement*{s}

Reads a tape record, and assigns the value returned to the specified variable. The length of the tape record is specified by the most recently executed T-ATT command. See the section on the THEN/ELSE Construct. As of release 2.2 this statement now works with streaming cartridge tape. The ELSE condition is taken if the tape unit is not attached, or an end-of-file is encountered.

**READU *arrayvar* FROM {*filevar*,} *idexp* . .
. . {LOCKED *statement*{s}} {THEN *statement*{s}} ELSE *statement*{s}**

Reads item from the optionally-specified file variable and assigns each attribute to an array element in the dynamic array designated in *arrayvar*. The READU statement is identical to the READ statement, except that the *group* is “locked”, preventing access to that group by any other process. Note that most other Pick implementations lock only the *item* during the read, and the *group* during an update. See the section on the THEN/ELSE Construct.

If the file variable parameter is not specified, the default file variable is used. See the section called “About Default File Variables” near the beginning of this section.

The ELSE condition is taken when the item is not on file. The THEN condition is taken when the item is read successfully.

As of release 2.2, the LOCKED clause is now supported in the syntax of this statement. This clause occurs before the THEN and/or ELSE clause(s) and specifies the statement(s) to execute if the group (or item in newer versions) is locked when the read is attempted. Earlier versions of generic Pick did not support this feature.

**READV *variable* FROM {*filevar*,} *idexp*, *attexp* . .
. . {LOCKED *statement*{s}} {THEN *statement*{s}} ELSE *statement*{s}**

Reads item from the optionally-specified file variable and assigns the value contained in the attribute number referenced in the attribute expression to the specified variable. See also: the READVU statement, the section on the THEN/ELSE Construct and the section called “About Default File Variables” near the beginning of this section.

The ELSE condition is taken if the item is not on file. The THEN condition is taken when the item is read successfully.

As of release 2.2, the LOCKED clause is now supported in the syntax of

this statement. This clause occurs before the THEN and/or ELSE clause(s) and specifies the statement(s) to execute if the group (or item in newer versions) is locked when the read is attempted. Earlier versions of generic Pick did not support this feature.

READVU *variable* FROM {*filevar*,} *idexp,attexp* . .
. . {LOCKED *statement*{*s*}} {THEN *statement*{*s*}} ELSE *statement*{*s*}

Reads item from the optionally-specified file variable and assigns the value contained in the attribute number referenced in the attribute expression to the specified variable. The READVU statement is identical to the READV statement, except that the *group* is “locked”, preventing access to that group by any other process. Note that most other Pick implementations lock only the *item* during the read, and the *group* during an update. See also the READV statement and the section on the THEN/ELSE Construct.

The ELSE condition is taken if the item is not on file. The THEN condition is taken when the item is read successfully.

As of release 2.2, the LOCKED clause is now supported in the syntax of this statement. This clause occurs before the THEN and/or ELSE clause(s) and specifies the statement(s) to execute if the group (or item in newer versions) is locked when the read is attempted. Earlier versions of generic Pick did not support this feature.

RELEASE

RELEASE *idexp*

RELEASE *filevar,idexp*

Releases the group locked with a previous MATREADU, READU or READVU statement, in the optionally-specified file variable. If the file variable and item-id expressions are both omitted, all groups currently locked by the current process are unlocked. If the file variable parameter is not specified, the default file variable is used. See the section called “About Default File Variables” near the beginning of this section.

REM *text*

Defines user-specified remarks. Causes the entire source line to be ignored by the compiler. The characters, * or ! may (and should) be substituted for the REM statement.

REM(*dividend,divisor*)

Calculates the remainder (modulo) of two numeric expressions. Same as the MOD function.

REPEAT

Terminates DO portion of LOOP . . WHILE or LOOP . . UNTIL statements. See the LOOP . . WHILE or LOOP . . UNTIL statements. The typical form at is:

```
LOOP {statement{s}} UNTIL condexp DO
    statement{s}
.
.
REPEAT
```

REPLACE(arrayvar,attrex,exp)

REPLACE(arrayvar,attrex,valex,exp)

REPLACE(arrayvar,attrex,valex,subvalex,exp)

Replaces the specified location within the dynamic array variable referenced by *arrayvar* with the value referenced in *exp* (expression). See also the INSERT function.

Note: A “- 1” may be specified as the *attrex*, *valex*, or *subvalex*. This causes the *exp* to be replaced as the last element in the respective location. An alternate method, using the dynamic array reference symbols is:

$$strex<arrayvar\{,valex\{,subvalex\}\}> = exp$$

Note that only the last form shown and the alternate method are acknowledged by the SMA standard.

RETURN

RETURN TO *statement.label*

Terminates an internal or external subroutine. Control resumes with the statement following the CALL or GOSUB command. The TO clause may only be used with an internal subroutine, and transfers control to the specified statement label. This is *not* recommended as it makes programs nearly impossible to figure out.

REWIND {THEN *statement{s}*} ELSE *statement{s}*}

Rewinds the magnetic tape on the attached unit. The ELSE condition is taken if the tape unit is not ready or attached with a T-ATT command. See the section on the THEN/ELSE Construct. As of release 2.2 this statement now works with streaming cartridge tape.

RND(*numexp*)

Generates a random number between 0 and the value specified in the numeric expression, minus one. For example:

PAY.INCREASE.PERCENTAGE = RND(8) + 1

The RND function in this example generates a random number between 0 and 7. One is added to the result to make sure that everybody gets at least some sort of an increase, making the range between one and eight.

RQM {*number-of-seconds*}

RQM {*untiltime*}

Activates the sleep function, placing process into an inactive mode until optional argument is satisfied. This statement terminates the current process' timeslice. Same as SLEEP statement. The sleep parameter may either be an integer number of seconds, or the desired time, in military format (delimited by colons). If no argument is indicated, the process will sleep for one second.

SELECT

SELECT *var* {TO *selectvar*}

SELECT *filevar* {TO *selectvar*}

Creates a list of item-id's from the file specified in the file variable parameter, allowing sequential access to each item in the file. If the file variable parameter is not specified, the default file variable is used. See also the READNEXT statement and the section called "About Default File Variables" near the beginning of this section. When used with the TO clause, the item list is assigned to the specified select variable.

SEQ(*exp*)

Converts an ASCII character to its numeric equivalent. The inverse of CHAR.

SIN(*numexp*)

Calculates the sine of the angle specified in the numeric expression, and returns the result in degrees.

SLEEP {*number-of-seconds*}

SLEEP {*untiltime*}

Activates sleep function, placing process into an inactive mode until optional argument is satisfied. This statement terminates the current process' timeslice. Same as the RQM statement. The sleep parameter may either be an integer number of seconds, or the desired time, in military format (delimited by colons). If no argument is indicated, the process will sleep for one second. Here are some examples:

```
NAPTIME = 3600  
SLEEP NAPTIME
```

```
UNTIL.WAKEUP.TIME = 8:00  
SLEEP UNTIL.WAKEUP.TIME
```

SPACE(*numexp*)

Generates a string of spaces whose length is equal to the value of the numeric expression.

SQRT(*numexp*)

Calculates the square root of the number defined in the numeric expression.

STOP

STOP *errmsg#*, "*parameter*" { "*parameter*" . . }

Terminates program. The *parameter*{s} are passed to the error message handler and displayed with the message stored in the message number indicated by "*errmsg#*". See also the ABORT statement. See ERRMSG section for available messages. Here are some examples:

```
OPEN "INVOICES" TO INV.FILE ELSE STOP 201, "INVOICES"  
READ CUST.ITEM FROM CUST.FILE, ID ELSE STOP 202, ID
```

STR(*exp*, *numexp*)

Generates a string of the variable expression repeated the number of times specified in numeric expression. See also the SYSTEM function for retrieval.

ing the current device width for substituting as the numeric expression. For example:

```
PRINT STR('***',79)
```

Prints 79 "***" (asterisks) at the current cursor (or print head) position.

SUBROUTINE {(*argumentlist*)}

SUBROUTINE *subroutinename* {(*argumentlist*)}

Defines a program as an external subroutine. This statement must appear as the first line in an external subroutine,⁸ and all arguments defined in the optional argument list must be delimited by commas (.). See also the CALL statement. All external subroutines should be cataloged prior to execution.⁹

SYSTEM(*exp*)

Interrogates the current value of pertinent system functions, as shown below:¹⁰

<i>Function</i>	<i>Description</i>
(0)	Relates to last tape-handling error generated by the READT, WRITET, WEOF or REWIND statements: 1 If not attached 2 Null variable 11 Record truncated
(1)	Returns a 1 if the system printer is on, meaning that a PRINTER ON statement has been issued or that the program was activated with a (P) option. Otherwise, a 0 is returned, meaning that output is being directed to the terminal
(2)	Returns the page width as defined by the TERM command
(3)	Returns the page length as defined by the TERM command
(4)	Returns the number of lines remaining to print on the current page, based on the current terminal characteristics previously defined with the TERM command (See <i>Note</i>)
(5)	Returns the current page number. SMA defines this as SYSTEM(6) (See <i>Note</i>)

⁸ Some versions require that this statement appear as the first *executable* statement, meaning that the program could contain comment lines before the SUBROUTINE line.

⁹ Some versions do not require this as long as the called program exists in the same file as the calling program.

¹⁰ Again, this function differs between various implementations. Consult your system documentation for differences.

<i>Function</i>	<i>Description</i>
(6)	Returns the current line counter. SMA defines this as SYSTEM(5). See note below
(7)	Returns the terminal type code, as defined by the TERM command
(8)	Returns the block size at which the tape was last attached. As of release 2.2 this function now works with streaming cartridge tape
(9)	Returns the current CPU millisecond count
(10)	Checks the current stack (STON) condition. Returns one (1) if the stack is on, or zero (0), if it is not on
(11)	Checks status of list function and returns one (1) if a list is active, or zero (0) if no list is active
(12)	Returns system time in milliseconds
(13)	Forces RQM (terminates timeslice) and returns a 1
(14)	Returns the number of bytes awaiting input in the input buffer
(15)	Returns verb option(s) in effect
(16)	Returns the number of levels of nested EXECUTE statements
(17)	Returns the ERRMSG item-ids from previous EXECUTE statement, separated by attribute marks
(100)	Returns release level, version and version date on Pick XT and AT only

Note: Arguments 4,5,6 may only be used in conjunction with the HEADING and FOOTING statements within PICK/BASIC.

TAN(*numexp*)

Calculates the tangent of the angle specified in degrees by numeric expression.

THEN/ELSE

In-line initiator. See the section at the end of PICK/BASIC on the THEN/ELSE Construct.

TIME()

Returns the current system time in internal format. Internal time is represented as an integer number indicating the number of seconds past midnight. Note that the parentheses following the function are required, and never contain any arguments. See also the DATE() function.

TIMEDATE()

Returns current system time and date in external format (hh:mm:ss dd mmm yyyy). Note that the parentheses following the function are required,

and never contain any arguments. See also the TIME() and DATE() functions.

TRIM(*strex*)

Removes all leading and trailing blanks from string expression, and leaves one blank wherever more than one blank was embedded within the string. For example:

```
STRING = " PICK POCKET GUIDE "  
STRING = TRIM(STRING)
```

After the TRIM, "STRING" would contain : "PICK POCKET GUIDE".

UNLOCK

UNLOCK *locknumber-expression*

Resets an execution lock, in the range 0 to 63. If no *locknumber* is specified, *all* locks set by the program are unlocked. See also the LOCK statement. Note that some implementations now support up to 128 locks.

UNTIL

In-line initiator in a FOR . . NEXT statement. Multi-line initiator in a LOOP statement. See FOR . . NEXT and LOOP . . WHILE statements.

WEOF {THEN *statement{s}*} ELSE *statement{s}*}

Writes an end-of-file mark to tape. See the section on the THEN/ELSE Construct. The ELSE condition is taken if the tape is not ready or attached. As of release 2.2 this statement now works with streaming cartridge tape.

WHILE

In-line initiator in a FOR . . NEXT statement. Multi-line initiator in a LOOP statement. See FOR . . WHILE and LOOP . . WHILE statements.

WRITE *arrayvar* ON {*filevar*,} *idexp*

Writes the item from the dynamic array specified in *arrayvar* into the specified file, using the item-id specified in the item-id expression. If the file variable parameter is not specified, the default file variable is used. See also

the WRITEU statement and the section called “About Default File Variables” near the beginning of this section.

WRITET *exp* {THEN *statement*{*s*}} ELSE *statement*{*s*}

Writes the value of the expression to tape. The ELSE condition is taken if the tape is not ready or attached. See the section on the THEN/ELSE Construct. See also the T-ATT command in the TCL section. As of release 2.2 this statement now works with streaming cartridge tape.

WRITEU *arrayvar* ON {*filevar*,} *idexp*

Writes the item from the dynamic array specified in *arrayvar* into the specified file, using the item-id specified in the item-id expression. If the file variable parameter is not specified, the default file variable is used. See also the WRITE statement and the section called “About Default File Variables” near the beginning of this section.

The WRITEU statement is identical to the WRITE statement, except that the group remains locked. See also the RELEASE statement.

WRITEV *exp* ON {*filevar*,} *idexp*, *attexp*

Writes the value of the expression into the attribute designated in the attribute expression parameter, using the item-id specified in the item-id expression parameter. If the file variable parameter is omitted, the default file variable is used. Note that this statement causes a READ to occur prior to the WRITEV, which accounts for the reason that this is inefficient. See also the WRITEVU statement and the section called “About Default File Variables” near the beginning of this section.

WRITEVU *exp* ON {*filevar*,} *idexp*, *attexp*

Writes the value of the expression into the attribute designated in the attribute expression parameter, using the item-id specified in the item-id expression parameter. If the file variable parameter is omitted, the default file variable is used. Note that this statement causes a READ to occur prior to the WRITEV, which accounts for the reason that this is inefficient. See also the WRITEV statement and the section called “About Default File Variables” near the beginning of this section.

The WRITEVU statement is identical to the WRITEV statement, except that the group remains locked. See also the RELEASE statement.

XTD(*exp*)

Converts value in expression from hexadecimal to decimal. See also the DTX function.

ABOUT PICK/BASIC USER EXITS

User exits are special functions that make direct references to assembler routines known as “modes”. They were added to the system to fulfill needs outside the standard boundaries of the PICK System prior to being replaced by extensions to the system. User exits have always been dangerous, and as long as they remain, will continue to be. *Extreme* caution is advised in using them. Caveat Emptor. The general form of user-exit is:

```
. . . = OCONV(argument,"user-exit")
```

For example:

```
WHO = OCONV(" ", "U50BB")
```

In other words, the user exits may be used in an assignment (on the right side of an equal sign) or after a PRINT statement. Note that many of the functions performed by user exits may now be achieved by using the EXECUTE statement with the appropriate TCL command or PICK/BASIC statement. The following is a list of user exits and their function:

<i>User Exit</i>	<i>Description</i>
U3060	Returns 8 hexadecimal characters as a value from supplied string. Used by the PASSWORD program to “encrypt” passwords
U1072	Activates “SORT” from PICK/BASIC. Sort of
U0079	Returns the same output as the “WHERE” verb from TCL
U3079	Converts PCB-FID (Primary Control Block frame-id either way, in an ICONV or OCONV statement
U307A	Sleep until specified military time. See the SLEEP or RQM statements in this section ¹¹
U407A	Sleep for specified number of seconds (See the SLEEP or RQM statements in this section)
U50BB	Returns the same output as the “WHO” command VERB from TCL
UF070	Returns the number of physical serial ports. (Release 2.2 and higher only)

¹¹ Advice: don’t use this. This is documented so that you can replace this with a valid statement if it is encountered in your code.

THE THEN/ELSE CONSTRUCT

The following statements make use of the PICK/BASIC “THEN/ELSE” construct:¹²

IF	LOCATE
LOCK	OPEN
MATREAD	MATREADU
READ	READNEXT
READT	READU
READV	READVU
REWIND	WEOF
WRITET	

Each of these statements are referred to as initiators of the THEN or ELSE clause that succeeds them. The possible syntactic structures are:

- 1) *initiator THEN statements ELSE statements*
- 2) *initiator THEN statements*
- 3) *initiator ELSE statements*

When THEN and ELSE clauses are both present, the THEN clause may be considered to be the initiator of the ELSE clause. All other characteristics are identical, except that an ELSE clause may succeed a THEN clause. For simplicity's sake, this will cover only the case of the THEN clause.

Within the THEN clause, the THEN token is the initiator, which requires a terminator. The clause may exist on one or more than one line. The nature of the terminator varies between these cases.

➤ Single-Line Form

If the THEN clause is complete in one physical line, its terminator is a <cr> or an ELSE token.

initiator THEN statement(s)<cr>

or

*initiator THEN statement(s) ELSE
statement(s)<cr>*

¹² Many implementations have added the THEN/ELSE construct to other statements, such as INPUT, WRITE, CLEARFILE and DELETE. Consult your system manuals for more details.

or

```
initiator ELSE statement{s}<cr>
```

The form for the single-line clause:

```
THEN statement; statement; . . . ;<cr>
```

or

```
THEN statement; statement; . . . ;ELSE<cr>
```

The syntax of the ELSE clause is the same as that of the THEN clause, except that it may not be followed by another ELSE clause.

➡ Multi-Line Form

If the THEN clause spans more than one physical line, the THEN token must immediately be followed by an EOL (End-Of-Line) character. This may be either a <cr> or a ; (semi-colon). In this case, the clause is terminated by END preceded by an EOL character.

```
initiator THEN<cr>
    statement{s}<cr>
    statement{s}<cr>
    statement{s}<cr>
END
```

```
THEN EOL statement EOL statement EOL . .
. . EOL END ELSE EOL statement EOL . .
. . statement END EOL
```

There must be one or more statements between the THEN and its terminator. If there are several statements, they must be separated by EOL characters. If the THEN clause is the one-physical-line clause, the EOL character must be a semi-colon. If the THEN clause spans more than one physical line, the EOL characters may be either (or both) <cr>'s or semi-colons.

The form of the multi-line clause is:

```
THEN EOL statement EOL statement EOL EOL END
EOL
```

or

```
THEN EOL statement EOL statement EOL . .
. . EOL END ELSE EOL statement EOL . .
. . statement END EOL
```

In this case, each EOL character may be either a <cr> or a semi-colon. This means that a multi-line clause may be contained in either one or more than one physical line. This case will normally appear as:

```
THEN<cr>
    statement<cr>
    statement<cr>
    statement<cr>
END<cr>
```

For program clarity, the statements are indented from the beginning of the initiator, and the THEN or END ELSE are outdented to the beginning of the initiator.


➡➡ Multiple END statements

Multi-line statements may have more than one END statement. The END statement then becomes the initiator for the ELSE condition as follows:

```
initiator THEN<cr>
    statement<cr>
    statement<cr>
    statement<cr>
END ELSE<cr>
    statement<cr>
    statement<cr>
    statement<cr>
END
```

Multiple END ELSE sequences may be used, provided that each ends with an END statement.

Access



This chapter is divided into four sections that are listed in the order that they appear:

- 1) The first section covers the general form of ACCESS commands and notes related issues such as using quotes and logical operators.
- 2) The second section provides an alphabetic listing of all ACCESS commands.
- 3) The third section provides an alphabetic listing of all ACCESS modifiers and connectives.
- 4) The fourth and final section provides an alphabetic listing of all the correlative and conversion codes.

➤ **About the Terms Used in This Section**

Please refer to the term glossary located at the back of the Pocket Guide. It contains definitions of the standard terms used in the PICK system, along with the special terms that were created during the production of this guide.

➤ **About Compatibility With the SMA Standard**

Some instructions in this section may not conform to the SMA standard. Those that are unique to the Pick AT/XT versions are indicated with a

compatibility index following the instruction. If no index is provided, the instruction conforms to the SMA standard.

GENERAL FORMAT OF ACCESS SENTENCES

>verb filename {itemlist} {seqlist} {sellist} {outlist} {modlist} {(options)}

STANDARD OPTIONS FOR ACCESS COMMANDS

- B** Suppresses linefeed after compile phase
- C** Activates COL-HDR-SUPP function
- D** Activates DET-SUPP function
- F** Specifies ejection of a page on each new item. Used in conjunction with COPY, LIST-ITEM, and SORT-ITEM verbs
- H** Activates HDR-SUPP (SUPP) function
- I** Activates ID-SUPP function
- N** Activates NOPAGE function, on output to terminal
- P** Directs output to system printer, via spooler
- Y** Shows ACCESS compiled string to the terminal, even if the P option is in effect¹
- Z** Shows ACCESS-compiled string to the printer, even if the P option is not in effect

NOTES ON THE USE OF ACCESS COMMANDS

- ☐ ACCESS commands may be issued from TCL, PROC or PICK/BASIC (using the EXECUTE command).
- ☐ The ACCESS verb must be the first word in the command and one file name must be specified, unless the USING modifier is present, in which case, two file names are needed.
- ☐ Any number of Attribute-Defining-Items may be requested for output by the command, provided all are defined in the dictionary of the file being accessed.
- ☐ Any number of modifiers and relational operators may be used in an ACCESS sentence, provided all are defined in the Master Dictionary.
Note: Only nine AND clauses are allowed in any ACCESS sentence.
- ☐ Each word or character in an ACCESS sentence must be delimited with a blank, unless it is a quoted string.
- ☐ Multiple sort keys are processed left to right.

¹ Not officially supported, but usually works.

USING QUOTES IN AN ACCESS SENTENCE

The use of single or double quote marks depends on the nature of the command. For referencing values (data elements), the string should be enclosed in double quotes. For example:

```
>LIST CUSTOMERS WITH NAME = "JESJ"
```

For referencing item-ids, the (item-id) string should be enclosed in single quotes. For example:

```
>LIST CUSTOMERS '100' '101' '102'
```

Both item-id strings and value strings may be included in an ACCESS statement. For example:

```
>LIST CUSTOMERS '100' '101' '102' WITH CONTACT = "[JON]"
```

This case, demonstrates the only time that Pick enforces the rules about single quotes surrounding *item-ids* and double quotes surrounding *value strings*.²

The backslash (\) character may also be used as a string delimiter. This is useful when the string being searched for contains both single and double quotes.

LOGICAL OPERATORS

Logical operators are used in the selection criteria process. This is the typical format:

```
WITH attribute-name operator . .  
. . "valuestring"
```

Here are some examples:

```
WITH STATE = "CA"  
WITH AMOUNT.DUE # "0"  
WITH BIRTHDAY < "1/1/62"  
WITH AGE > "21"
```

² Some implementations have removed the necessity of placing item-ids in quotes at all. You may want to test this.

Note that each operator has an "English" equivalent which may be used interchangeably with the "symbol" form. When no operator is present, the default assumed is an "equal to" (=). Here are some examples:

WITH STATE EQ "CA"
WITH AMOUNT.DUE NOT "0"
WITH BIRTHDAY BEFORE "1/2/62"
WITH AGE GT "21"

SPECIAL SYMBOLS AND ALTERNATE MEANINGS

<i>Symbol/Word</i>	<i>Meaning</i>
=, EQ	Equal to
#, NE, NO, NOT	Not equal to
>, GT, AFTER	Greater than
<, LT, BEFORE	Less than
>=, GE	Greater than or equal to
<=, LE	Less than or equal to
&, AND	Logical boolean connectives

LOGICAL CONNECTIVES: AND AND OR

Logical connectives are boolean operators used to connect two or more selection criteria phases in an ACCESS sentence. When two phases are connected with an "OR", *either* phase may be true for data to be selected for processing. For example:

WITH CITY "RICHMOND" **OR** WITH STATE "CA"

When an "AND" clause connects two phases, *both* phases have to be true for data to be affected. For example:

WITH CITY "RICHMOND" **AND** WITH STATE "CA"

When the connective between two phases is omitted, the default connective is an OR. For example:

WITH CITY "RICHMOND" WITH STATE "CA"

The AND clauses take precedence over OR clauses, and a maximum of nine AND clauses may be specified.

THROWAWAY CONNECTIVES

Throwaway connectives are words that are used to provide continuity to a report, but are actually ignored by the ACCESS language. These words reside in your MD and the normal set includes the following

A	FILE	OF
AN	FOR	THE
ANY	ARE	IN
ITEMS		

This list of available throwaway connectives may easily be expanded by copying any one of the above items in the MD to new items, as in the following example:

>COPY MD A A A A<cr>
TO:DISPLAYING SHOWING FIELD ATTRIBUTE<cr>

After adding the new items, the following sentence is now possible:

```
>SORT THE CUSTOMER FILE SHOWING THE NAME FIELD<cr>
```

Where “THE”, “FILE”, “SHOWING”, and “FIELD” are all ignored, but would provide for a more grammatically correct sentence. This makes it easier for tailoring user vocabularies to foreign languages, like Texan for example.

THE USING CONNECTIVE

The general format for the USING connective is:

USING *DICT filename*

Specifies the file dictionary to access for attribute definition items. For example:

>LIST CUSTOMERS USING DICT SUPPLIERS

STRING SEARCHING

When used in conjunction with the “WITH” modifier, data elements may be searched for the occurrence of a string of characters, as illustrated in the following examples:

... WITH *attribute operator* "[value]"

Searches entire element for occurrence of “value”. Any string of characters may precede or follow “value”.

. . . WITH *attribute operator* “[value”

Only returns data when the element being searched “ends” with “value”, that is, any string of characters may precede “value”, but the element must end with “value”.

. . . WITH *attribute operator* “[value”]

Only returns data when the element being searched “begins” with “value”, that is, any string of characters may follow “value”, but the element must begin with “value”.

. . . WITH *attribute operator* “[. . . ^ . . .]”

Wildcard search. Used for searching for any characters within strings. For example:

. . . WITH NAME = “[SM^TH”

Would retrieve any item containing at least five characters, the first two of which begin with “SM”, followed by any character, and ending with “TH”. In other words, this would retrieve “SMITH” as well as “SMYTH”.

MULTIPLE STRING SEARCHES

When applying multiple selections against a single attribute value, it is not necessary to build a separate selection criteria phase for each, as in the following examples:

```
>LIST CUSTOMERS WITH STATE = “CA” OR WITH  
STATE = “AZ” OR WITH STATE = “VA”
```

This could also be stated as:

```
>LIST CUSTOMERS WITH STATE “CA” “AZ” “VA”
```

Here are some examples of a Wildcard Search:

```
>LIST CUSTOMERS WITH NAME = “I^M”
```

Searches the file for items whose attribute, defined by NAME, contains a string beginning with “I”, followed by any character, and ending with “M”.

```
>LIST CUSTOMERS WITH NAME = “A^^”
```

Searches attribute for values containing a string beginning with “A” and followed by two characters. Any two characters following the “A” will meet the selection criteria.

GENERAL FORMAT OF ATTRIBUTE-DEFINING-ITEMS

<i>Attribute Number</i>	<i>Description</i>
000 item-id	The name of the item
001 type	Attribute definition type A - Attribute definition S - Synonym attribute definition X - Protected attribute (Skipped on implicit reports)
002 amc	Attribute Mark Count
003 header	Optional column header
004 associative	Associated attribute
005 Not used	
006 Not used	
007 conversion	Any valid conversion. Handled at post processing phase (print time)
008 correlative	Any valid correlative. Handled at pre-processing phase (select/sort time)
009 justify	Output column justification L Left justified R Right justified T Left with wraparound on blank U Unconditional left
010 width	Output column width maximum length

<i>Line</i>	<i>NAME</i>	<i>Function/Description</i>
1	D/CODE	Pointer (A,S,X)
2	A/AMC	Attribute Mark Count
3	S/NAME	Column Heading
4	S/AMC	Associated attributes.
5	L/RET	Reserved for file pointers.
6	L/UPD	Reserved for file pointers.
7	V/CONV	Conversion
8	V/CORR	Correlative
9	V/TYP	Justification (L,R,T,U)
10	V/MAX	Maximum output width.

ACCESS VERBS

The following are the ACCESS verbs in alphabetical order.

**>CHECK-SUM *filename* {*itemlist*} {*attrname*} . .
. . {*sellist*} {*modlist*} {(options)}**

Calculates the hexadecimal checksum for either a specified attribute, or if no attribute is specified, the entire item in the specified filename.

>COPY-LIST *itemlist {(options)}
TO:({*filename*} {*listname*})**

Copies a previously saved listname to either a new name, a new filename, or the specified output device. Defaults to POINTER-FILE for filename. The following are the *options* for the COPY-LIST command:

- D Deletes item from source file after copy
- N Activates NOPAGE function on output to terminal
- O Overwrites duplicate itemname(s)
- P Copies list to system printer, via spooler
- T Copies list to terminal
- X Outputs list in hexadecimal format

>COUNT *filename* {*itemlist*} {*sellist*} {*modlist*} {(options)}

Outputs the number of items that meet the selection criteria as determined by the *itemlist* and *sellist* specifications.

>DELETE-LIST *itemlist**

Removes the specified list name(s) from the POINTER-FILE. Note that this is a modified TCL-2 type command, meaning that one may issue the command: >DELETE-LIST *, which means ALL saved-lists. This has the effect of leaving your system quite listless.

>EDIT-LIST *listname* {(Z)}

Retrieves specified list name and enters editor processor. Refer to the EDITOR section for the available commands and syntax requirements. This is the only option:

- Z Suppresses the “Top” and “EOI” messages.

>FILE-TEST *filename* {*itemlist*} {*sellist*} {(options)}

In the SMA standard document, this verb replaces the HASH-TEST and ISTAT verbs formerly used to analyze the statistics and distribution of items in a file. The statistics that it produces include: the count of items, the total number of bytes, and the average bytes per item. Additionally, the SMA standards document indicates that the information *may* also include: a hashing structure illustrated with a histogram, the average number of items per group and standard deviation, and the average number of bytes per group. The relevant option is:

S Suppresses detail; only shows statistical data.

Compatibility: SMA

>GET-LIST {*filename*} *listname*

Retrieves the *listname* from specified file, or defaults to POINTER-FILE if no *filename* is specified.

>HASH-TEST *filename* {*itemlist*} {*sellist*} {(options)}

TEST MODULO:modulo <cr>

Outputs a histogram showing the item hashing distribution of all items meeting the selection criteria with a test modulo, entered by the operator. See also the ISTAT command. The option is:

S Suppresses display of histogram.

>ISTAT *filename* {*itemlist*} {*modlist*} {(options)}

Outputs a histogram showing the item hashing distribution, based on specified filename's current modulo and the optionally specified selection criteria. See also the HASH-TEST command. The option is:

S Suppresses display of histogram.

>LIST *filename* {*itemlist*} {*sellist*} {*modlist*} {*outlist*} {(options)}

Outputs specified items, in order of appearance in the *filename*.

>LIST-ITEM *filename* {*itemlist*} {*sellist*} {*modlist*} {(options)}

Outputs items, in COPY format.

>LIST-LABEL *filename* {*itemlist*} {*sellist*} {*outlist*} {*modlist*} {(options)}

Outputs formatted labels from specified *filename* in order of item appearance in file name. Upon pressing the carriage return the prompt character, ?, appears and expects the parameters in this format:

?columns,rows,skip,indent,size,space,C

The following is a description of the parameters:

<i>Parameter</i>	<i>Description</i>
columns	Number of labels across page
rows	Number of lines requested to print on each label
skip	Number of lines (vertically) between labels
indent	Number of spaces from left margin to beginning output column
size	Number of print positions per label
space	Number of spaces (horizontally) between labels
C	Compresses null values
Note:	Each of the parameters must be separated by commas during input

After the entry of the above parameters, if the INDENT parameter is non-zero, the prompt character, ?, appears for the number of times indicated in the rows parameter. (i.e. if rows was 4, expect 4 question marks) These prompts (?’s) are for the entry of *headers* that print in the INDENT area to the left of the first row of labels. For example:

```
>LIST-LABEL VENDORS NAME ADDRESS CSZ (C<cr>
?2,3,1,0,30,2,C<cr>
```

>QSELECT *filename itemlist {(attrnum)}**

Creates a select list item from the specified *filename* and *itemname* entered. This is similar to the GET-LIST command. If the optional *attrnum* (attribute number) is specified, the list is created from the multivalues contained in the specified attribute number.

Note that this is actually a TCL-II command, but is included here for its use in building lists.

>REFORMAT *filename* {*itemlist*} *outlist* {*modlist*}
FILE NAME:*destination-file* or TAPE<cr>

Data elements that are defined in the specified *filename* are put into another *filename*, by the order of appearance in the *outlist*. Note that the first entry in the *outlist* is used as the item-id for the destination file. Each subsequent entry in the *outlist* becomes an attribute in the destination item, in the order of specification in the *outlist*.

Upon execution of this command, the prompt, FILE NAME:, appears on the screen. Either a filename may be entered, directing the output to the specified filename, or the literal, TAPE, which directs the output to the attached magnetic tape.

>S-DUMP *filename* {*itemlist*} {*seqlist*} {*sellist*} . .
. . {*modlist*} {HEADING "*text*" } {(*options*)}

Copies to tape all item names from the specified *filename* that meet the specified selection criteria, in order of the specified sort key(s). The HEADING modifier specifies the tape label to write to the tape prior to the data. The following is a list of options:

- H Suppresses the tape label
- I Suppresses the display of item-id's as they are dumped

>SAVE-LIST (*filename*) *listname*

Saves the *listname* in specified *filename*. The default file name is POINTER-FILE.

>SELECT *filename* {*itemlist*} {*sellist*} {*outlist*} {*modlist*}

Retrieves all items from specified *filename* that meet selection criteria, in order of their appearance in the file name, and creates a list that either may be saved for future use or acted on immediately. The *outlist* parameter specifies the attribute definition item(s) from which values are to be extracted for creation of the list.

>SORT *filename* {*itemlist*} {*seqlist*} {*sellist*} {*outlist*} {(*options*)}

Outputs specified file name, in order of specified sort key(s).

>SORT-ITEM *filename* {*itemlist*} {*seqlist*} {*sellist*} {*modlist*} {(options)}

Outputs items, in COPY format.

**>SORT-LABEL *filename* {*itemlist*} {*sellist*} . .
. . {*seqlist*} {*outlist*} {*modlist*} {(options)}**

Outputs formatted labels from specified *filename* in order of specified sort key(s). Upon pressing the carriage return the prompt character, ?, appears for the entry of the following label parameters:

?columns,rows,skip,indent,size,space,C

Here is a description of the parameters:

<i>Parameter</i>	<i>Description</i>
columns	Number of labels across page
rows	Number of lines requested to print on each label
skip	Number of lines (vertically) between labels
indent	Number of spaces from left margin to beginning output column
size	Number of print positions per label
space	Number of spaces (horizontally) between labels
C	Compresses null values
Note:	Each of the parameters must be separated by commas during input.

After the entry of the above parameters, if the INDENT parameter is non-zero, the prompt character, ?, appears for the number of times indicated in the rows parameter. (i.e. if rows was 4, expect 4 question marks) These prompts (?)s are for the entry of headers that print in the INDENT area to the left of the first row of labels. For example:

```
>SORT-LABEL INVENTORY BY PART.# PART.# DESC PRICE (P<cr>
?3,4,1,0,30,2,C<cr>
```

**>SREFORMAT *filename* {*itemlist*} {*outlist*} {*seqlist*} {*modlist*}
FILE NAME:*destination-file* or TAPE**

Inserts data elements defined from the specified *filename* into another file name, in the order of the appearance in the *outlist*, after sorting the data by

the specified sort key(s). Note that the first entry in the outlist is used as the item-id for the destination file. Each subsequent entry in the outlist becomes an attribute in the destination item, in the order of specification in the outlist.

Upon execution of this command, the prompt, FILE NAME:, appears on the screen. Either a file name may be entered, which directs the output to the specified file name, or the literal — TAPE — which directs the output to the attached magnetic tape.

>SSELECT *filename* {*itemlist*} {*sellist*} {*seqlist*} {*outlist*} {*modlist*}

Retrieves all items from specified *filename* that meet selection criteria, in order of the specified sort key(s) and creates a list that may be saved for future use or acted on immediately. The *outlist* parameter specifies the attribute definition item(s) from which values are to be extracted for creation of the list.

>STAT *filename* {*itemlist*} {*sellist*} {*modlist*} {*attrname*} {(options)}

Outputs item count and an average value of all items that meet the specified selection criteria.

>SUM *filename* {*itemlist*} *attrname* {*sellist*} {*modlist*} {(options)}

Outputs total of the specified *attrname* for all item names meeting specified selection criteria. The default attribute name is zero (0) (item-id), if the attribute name is omitted.

**>T-DUMP *filename* {*itemlist*} {*sellist*} {*modlist*} . .
. . {HEADING "text"/options} {(options)}**

Copies to tape all *itemnames* that meet specified selection criteria. The HEADING modifier allows specifying the contents of the tape label written to the tape prior to the dump. The following are the options for this command:

- H Suppresses the tape label
- I Suppresses the display of item-id's as they are dumped

>T-LOAD *filename* {*itemlist*} {*sellist*} {*modlist*} {(options)}

Copies from tape to specified *filename* all items that meet specified selection criteria. The following is a list of options:

- I Suppresses listing item names to terminal
- O Overwrites existing items, when duplicate item-id's exist

MODIFIERS AND CONNECTIVES

Modifiers alter the normal operation of an ACCESS sentence and add special functions. A list of modifiers can be produced with the LISTCONN command.

BREAK-ON *attrname* "{text{'options'} . . .}"

Specifies creation of a logical and visual "break" in output data when the value in the current output field is different from the previous value. If the TOTAL modifier is specified, subtotals are also output. The *text* parameter, if specified, becomes a label that replaces the default literal *** normally output in the column in which the control break occurs. The following is a list of options:

- B Outputs the value causing the break in either the HEADING or FOOTING output field where the B option is found in the HEADING or FOOTING option string
- D Suppresses the break data line if only one detail line has been output since the last control break
- L Suppresses blank line preceding break data line
- N Resets the page counter to one
- P Forces form feed after output of data that caused control break
- R Outputs any occurrence of one or more control break lines at the end of the page, rather than at the top of the next page
- U Underlines all TOTAL fields
- V Outputs the value causing the control break in the BREAK-ON label
- " Inserts a single quote in text

Here are some examples:

```
>SORT INVOICES BY DATE BREAK-ON DATE . . .  
  
>SORT SALES BY SALESMAN BREAK-ON SALESMAN  
"BP" TOTAL INVOICE.AMOUNT TOTAL  
QUANTITY.ORDERED PRODUCT HEADING "LC'DETAIL  
SALES FOR 'B' PAGE 'PNL'"
```

BY *attrname*

Specifies *attrname* as sort key.

BY-DSND *attrname*

Specifies *attrname* as sort key, in descending sequence.

BY-EXP *attrname* {*explosionlimiter*}

Specifies multivalue *attrname* as sort key. The *explosion limiter* specifies that output only occurs on data elements meeting the print limiting criteria. For example:

```
>SORT CUSTOMERS BY-EXP INV.ID
```

or

```
>SORT CUSTOMERS BY-EXP INV.ID = "S100101"
```

BY-EXP-DSND *attrname* {*explosionlimiter*}

Specifies multivalue *attrname* as sort key, in descending sequence. The *explosion limiter* specifies that output only occurs on data elements meeting the print-limiting criteria.

COL-HDR-SUPP (or "C" option)

Identical to HDR-SUPP, but also suppresses column headings above output columns.

DBL-SPC

Outputs data on page with double spacing.

DET-SUPP (or "D" option)

Suppresses display of detail data. Typically used in conjunction with the TOTAL or BREAK-ON modifiers.

DICT

Causes ACCESS to reference dictionary level of specified filename. For example:

```
>SORT-ITEM DICT CUSTOMERS WITH *A2 = "A""S""X" BY *A2
```

EACH *attrname* {*operator*} {"*value*"}

Specifies that each value in a multivalued attribute must meet the specified selection criteria in order to be eligible for processing. Same as EVERY modifier. Used in the form: WITH {NOT} EACH . . .

EVERY *attrname* {*operator*} {"*value*"}

Specifies that each value in a multivalued attribute must meet the specified selection criteria in order to be eligible for processing. Same as EACH modifier. Used in the form: WITH (NOT) EVERY . . .

FOOTING "{*text*} {*options*} . . .}"

Specifies text to output at the bottom of each output page. Multiple options may be enclosed in the same set of single quotes. For example:

```
FOOTING "'LC'Page 'PN' Printed at 'TL'"
```

The following is a list of options:

- B Inserts value that caused break condition in FOOTING text. The B option must be included in the BREAK-ON specification
- C Centers output according to current page width
- D Outputs (external) system date
- F Outputs filename
- Fn Outputs filename left justified in a field of "n" blanks
- L Performs linefeed/carriage return
- P Outputs page number, right justified in a field of four blanks
- PN Outputs page number, left justified
- Pn Outputs page number, left-justified in a field of "n" blanks
- T Outputs (external) system time and date
- " Outputs one single quote

GRAND-TOTAL "{*text*} {*options*} . . .}"

Outputs specified text in place of the standard literal string *** normally output on the last total line. The following is a list of options:

- L Suppresses blank line preceding break data line
- P Forces form feed after output of data that caused control break
- U Underlines all TOTAL fields
- " Inserts a single quote in text

HDR-SUPP (or "H" option)

Suppresses display of the default Top of Page heading (system time/date and page number) and the number ITEMS LISTED message on output. Also the same as the "SUPP" modifier.

HEADING “*{text}* *{options}* . . . ”

Specifies text to output at the top of each output page. Multiple options may be enclosed in the same set of single quotes. For example:

```
HEADING "'LC'Page 'PN' Printed at 'TL'"
```

Here is a list of available options:

- B Inserts value that caused break condition in FOOTING text. The B option must be included in the BREAK-ON specification
- C Centers output according to current page width
- D Outputs (external) system date
- F Outputs filename
- Fn Outputs filename left justified in a field of “n” blanks
- L Performs linefeed/carriage return
- P Outputs page number, right justified in a field of four blanks
- PN Outputs page number, left justified
- Pn Outputs page number, left-justified in a field of “n” blanks
- T Outputs (external) system time and date
- “ Outputs one single quote

ID-SUPP (or “I” option)

Suppresses the output of item-ids.

IF (NOT) {EACH} *attrname* {operator} {“*value*”}

Specifies that the selection criteria must be present and true for data to be accepted for processing. Same as WITH modifier.

The NOT modifier indicates a test for the absence of *attrname value*. The value string may be preceded by a logical operator and must be enclosed within double quotes.

LPTR (or “P” option)

Directs output to system printer, via spooler.

NOPAGE (or “N” option)

Activates NOPAGE function on output to terminal, indicating that no carriage return is required between pages of display.

ONLY *filename*

Suppresses the output of implicit attribute definition items, and outputs only a list of item-id's. The ONLY modifier must precede the *filename* specification.

SUPP (or “H” option)

Suppresses display of the default Top of Page heading (system time/date and page number) and the number ITEMS LISTED message on output. Also the same as the “HDR-SUPP” modifier.

TAPE

Indicates that data will be retrieved from the currently attached magnetic tape or diskette, rather than the actual data level of the file. The tape must have been created in a T-DUMP format and must be positioned to the beginning of the appropriate (tape) file. The attribute definition items specified in the sentence are retrieved from the dictionary level of the specified filename.

This modifier may be used only with ACCESS verbs that do not sort prior to output, which includes the following:

COUNT	ISTAT
LIST	LIST-LABEL
REFORMAT	SELECT
STAT	SUM

TOTAL *attrname* {“*total-limiter*”}

Accumulates columnar total for specified attribute definition item. The optional *total-limiter* functions like the “value string” portion of the selection criteria process, limiting totals to values that pass the specified selection criteria.

USING {DICT} *filename*

Specifies the file dictionary or data section to access for attribute definition items or data. For example:

```
>LIST CUSTOMERS USING DICT SUPPLIERS
```

WITH {NOT} {EACH} *attrname* {operator} {"value"}

Specifies that the selection criteria must be present and true for data to be accepted for processing. Same as IF modifier.

The NOT modifier indicates to test for the absence of *attrname* value. The value string may be preceded by a logical operator and must be enclosed within double quotes.

>LIST WITHIN *filename* 'item-id' {(options)}

>COUNT WITHIN *filename* 'item-id' {(options)}

Retrieves a list of all the items that are sub-items of the specified *item-id*. This is used to explode one attribute containing one or more multivalues that are item-ids within the same file, like in Bill Of Material explosions. The file definition item (D-pointer) must contain a V code in attribute 8 in the form: V;;n where "n" indicates the AMC to explode.

WITHOUT *attrname*

Same as WITH NOT or WITH NO.

CORRELATIVES AND CONVERSIONS

➤➤➤ About Correlatives

Correlatives are special processing codes used to define an action to be taken on a specified attribute or set of attributes during the pre-processing phase of an ACCESS process, which is before data is selected and/or sorted.

➤➤➤ About Conversions

Conversions are also special processing codes used to define an action to be taken on a specified attribute or the results of a correlative during the post-processing phase of an ACCESS process, which is just before the data is output, printed or displayed.

FORMAT OF ATTRIBUTE-DEFINING-ITEMS

The following is a table showing the general format of Attribute-Defining-Items

<i>Format</i>	<i>Explanation</i>
001 A,S,X	Attribute definition type <i>A</i> Attribute definition <i>S</i> Synonym attribute definition <i>X</i> Protected attribute. Skipped on implicit reports
002 AMC	Attribute Mark Count
003 S/NAME	Optional column header
004 ASSOCIATIVE	Defines associated attribute(s)
005 Not used	
006 Not used	
007 CONVERSION	Postprocessing. Occurs to data immediately prior to being output
008 CORRELATIVE	Preprocessing. Occurs to data before sorting and/or selection criteria
009 L,R,T,U	Output column justification <i>L</i> Left <i>R</i> Unconditional right <i>T</i> Left with wraparound on blank <i>U</i> Unconditional left
010 MAX	Output column width maximum length

This table lists the correlatives and conversions that are described in the rest of this chapter:

<i>Correlative</i>	<i>Conversion</i>	<i>Line(s)</i>
A	Algebraic expression	7,8
C	Controlling attribute	4
C	Concatenation	7,8
D	Dependent Attribute	4
D	Date conversion	7,8
F	Mathematical Function	7,8
G	Group Extract	7,8
L	Length	7,8
M	Mask function	7,8
MC	Mask Characters	7,8
MD	Mask decimal (same as MR)	
ML	Mask, left-justified	7,8
MP	Mask Packed Decimal	7,8
MR	Mask, right-justified (Same as MD conversion)	7,8

<i>Correlative</i>	<i>Conversion</i>	<i>Line(s)</i>
MS	Mask Sequence	8
MT	Mask Time	7,8
MX	Mask Hexadecimal	7,8
MY	Mask Hexadecimal compression	7,8
P	Pattern match	7,8
R	Range	7,8
S	Substitution	7,8
T	Text Extraction	7,8
T	File Translation	7,8
U	User Mode Exit	7,8
V	WITHIN sublist code	7

➤➤➤ About SMA Standards

Most of the correlatives and conversion codes contained in this section remain compatible with the SMA standard. Those that do not are specifically noted.

The following are descriptions, in alphabetical order, of the conversions and correlatives:

A{n}{;}expression{s}

Recursive algebraic function for performing arithmetic and relational operations on specified operands. A correlatives are parsed into F correlatives by the ACCESS compiler. The functional operators and operands for the A correlative are the same as that of the F correlative, except as noted.

The *n* argument is optionally used to indicate the number of decimal digits (in the range 0–4) to retain during calculations involving a mixture of whole numbers and numbers with implied decimals, along with MR conversions in the body of the function.

The expressions formed are like BASIC expressions and consist of operands, operators, conditional statements and special functions combined together to yield a single resulting value.

➤➤➤ Operands of the A-correlative

n

An integer reference to an Attribute Mark Count. For example:

A3*5

Multiplies the value in attribute 3 by the value in attribute 5.

n(corr/conv{[corr/conv . . .]})

An integer reference to an AMC, which may be immediately passed through any other valid conversion or correlative, except another A or F correlative. Multiple correlatives or conversions may be used when each is delimited by a value mark. For example:

A1(T1,40)MCT)

Retrieves attribute 1 of the current item, extracts the first forty characters, and converts each word to upper- and lowercase.

9998

Used as an AMC specification. Returns the sequential item counter.

9999

Used as an AMC specification. Returns the size of the item in bytes.

“alphanumeric string”

Any quoted text or numeric string, which is treated as a literal. The following is an example of Literal:

A"Attention: ":1

Concatenates the literal, “Attention: “, to the contents of attribute 1 of the current item.

An example of a Numeric Constant:

A3*“12”

Multiplies the value of attribute three of the current item by the constant “12”.

N(attrname){(corr/conv . . .)}

Allows a reference to any valid attribute defining item in the dictionary of the file being accessed. Note that the string returned with this operand may subsequently be passed through one or more other valid conversions or correlatives (except another A or F correlative) as long as each is separated by a value mark. This is where recursion occurs, as the attrname being referenced in the correlative may in fact reference an item which also contains an A correlative. Here are some examples:

AN(CITY):", "::N(STATE):" "::N(ZIP)

Concatenates the contents of the attribute referred to by the item, "CITY", with a comma and a blank, then joins the value of the attribute referred to by
Allows a reference to any valid attribute defining item in the dictionary of

AN(CITY):", "::N(STATE)(TSTATES;C;;1)" "::N(ZIP)

Like before, but this time the value in the "STATE" attribute is "translated" to the STATES file, returning attribute 1 of the corresponding item.

D

Returns the system date in internal form. For example:

A(D-N(INVOICE.DATE))

Subtracts the value of INVOICE.DATE from the current system date, which determines the number of days that have elapsed since the date of the invoice.

NB

Returns the number of the break level. Note that there is a maximum of 255 break levels and that the 255th level is reserved for the grand total.

ND

Returns the number of detail lines. Used only on line 7 (conversion). Note that this may abort if the file contains more than 65,000 (about 64K) items. On some versions, it may crash with 32K items.

NI

Returns the number of the item being processed.

NV

Returns the number of the multi-value being processed.

NS

Returns the number of the Sub-value being processed.

T

Returns the system time in internal format, meaning the number of seconds past midnight.

➤➤ Arithmetic Operators

+ add * multiply

– subtract / divide

➤➤ Concatenation Operator

: (colon)

➤➤ Arithmetic Functions

R(expression1,expression2)

Returns the remainder of expression1 divided by expression2.

S(expression)

Sums multivalued results.

Substring Referencing:

strexp[begexp,lenexp]

Substring function. Used to extract a fixed number of characters from a string expression. The “begexp” (beginning position expression) specifies the beginning character position and the “lenexp” (length expression) specifies the length, or number, of characters to retrieve. The begexp and lenexp may be quoted numbers or any expressions which derive numeric results. For example:

A“(‘:N(PHONE)[‘1’,‘3’]:)’ ‘:N(PHONE)[‘4’,‘3’:N(PHONE)[‘7’,‘4’]

This takes a phone number stored as “7145551212” and outputs it as “(714) 555–1212”.

➤➤ Relational Operators and Effects

Relational operators produce a numeric result. These results are either 1 (one) for a true condition or 0 (zero) for a false condition.

= Equal to

If *operand1* = *operand2* then 1 else 0

Not equal

If *operand1* # *operand2* then 1 else 0

< Less than

If operand1 < operand2 then 1 else 0

> Greater than

If operand1 > operand2 then 1 else 0

<= Less than or equal to

If operand1 <= operand2 then 1 else 0

>= Greater than or equal to

If operand1 >= operand2 then 1 else 0

Celement{;element . . .}

Celement{/}

Concatenates any combination of elements. Each element may be delimited by a character to appear on output. The semi-colon (;) has a special meaning; it does NOT appear on output. The following is a list of elements:

- ☐ Numeric constants, enclosed in quotes.
- ☐ Attribute number reference by its numeric Attribute Mark Count.
- ☐ Alphanumeric character strings. When delimited by semi-colons, strings must be enclosed in quotes. Otherwise, alphanumeric strings are not separated from the other entries by semi-colons (;).

The asterisk (*) is a special character used to concatenate the current data value. For example:

C;3;" "4;" "5

Concatenates the value of attribute three of the current item with a comma and a blank, followed by attribute four, another blank, then attribute five.

C;attrnum{;attrnum . . .}

Controlling attribute function. Defines the control portion of a Control-Dependent attribute pair. Each attribute number specified in the "attrnum" parameter(s) must contain the dependent (D) code and attribute number of the controlling attribute. Note that this function belongs in attribute 4 of the item. For example:

C;4;5;6

Indicates that the current attribute "controls" the values in attributes four, five and six. Note that each of these three attributes will need to indicate in the attribute defining item that they are dependent upon this attribute.

D;attrnum

Dependent attribute function. Defines the dependent portion of a Control-Dependent attribute pair. The attribute number specified in the *attrnum* parameter must contain the controlling (C) code and attribute number of the controlling attribute. Note that this function belongs in attribute 4 of the item.

D(year.length){skipdel #skips}{outsep}

Date conversion function. Converts value from internal to external date format. The *year.length* parameter specifies the length of the year field. It must be between 0 and 4. The default length is the 4-digit year format.

The parameters specified in *skipdel* and *#skips* are functionally equivalent to the G (Group extract) correlative. They define the position of the date value portion in a multi-part data value and for the most part, are unused.

If included, the *outsep* (output separation character) parameter specifies the character that delimits the month, day and year values. The output order is 2-digit month, 2-digit day and the year. The number of year digits depends on the *year.length* parameter.

If the *outsep* parameter is omitted, the format is 2-digit day, 3-digit month abbreviation and year. Again, the number of year digits depends on the *year.length* parameter. Here are some examples of Date Conversions:

<i>Conversion</i>	<i>Format of Output</i>
D	dd mmm yyyy
D2—	mm-dd-yy
D—	mm-dd-yyyy
D0	dd mmm
D0—	mm-dd
DD	dd (numeric Day of month)
DI ³	nnnn (internal format)
DJ	nnn (Julian date)
DM	mm (numeric month)
DMA	xxxx (alphabetic month)
DQ	n (numeric quarter)
DW	n (numeric day of week)
DWA	xxxx (alphabetic day of week)
DY	yyyy (four-digit year)
D2Y	yy (two-digit year)

³ Note that the DI conversion is the only conversion which converts data from its external to its internal format in a dictionary. This is used in the unlikely case of having dates stored on disk in external format. In such cases the DI may be used as the correlative (attribute 8) during the preprocessing phase, and any other date conversion code may be used as the output conversion (attribute 7).

F(*n*);*element*{*element* . . .}

Compatibility: Pick (see also “FS” correlative)

Mathematical function for performing arithmetic and relational operations on specified operands. The F correlative is a stack-oriented function, requiring the statement syntax to adhere to a classic Reverse Polish Notation format. The push-down stack has 7 possible entries.

Unless otherwise noted, all of the operands and operators of the F-correlative are compatible with the SMA standard FS correlative. Here is an example of Post-fix Polish Notation:

F;3;4;*

This example instructs ACCESS to take the value of attribute three and multiply it by the value of attribute four.

All operands push a specific value on the top of the stack (STACK1). A push repositions all stack entries 1 position down. The entry in STACK7 is lost on an operand push.

➤ Operands of the F-correlative⁴

n{R}{(*conversion*)}

Numeric reference to an Attribute Mark Count to push on the stack. The “conversion” represents any legal ACCESS conversion.

“*alphanumeric string*”

Any quoted text or numeric string, which is treated as a literal. The following is an example of Literal:

F;"Attention: ";1;:

Concatenates the literal, “Attention: ”, to the contents of attribute 1 of the current item. The following is an example of Numeric Constant:

F;3;"12";*

Multiplies the value of attribute three of the current item by the constant “12”.

nR{R}

Optional repeat code for multivalues. The *n* indicates an AMC reference. The optional *R* parameter specifies repeat code for subvalues.

⁴ Note that all but the “R” (Repeat code) and the “LPV” (Load Previous Value) operands are available with the A-correlative.

Cn

Constant (alternate method). The n parameter may be any numeric string, or an alphanumeric string enclosed in quotes.

D

Returns the system date in internal form.

T

Returns the system time in internal form.

➡ Operators

Most operators act on the first two entries in the stack (STACK1 and STACK2). These entries are both popped off the stack, following execution. The result is then pushed back on top of STACK1.

➡ Mathematical Operators & Effects

Addition: $STACK1 = STACK2 + STACK1$
Subtraction: $STACK1 = STACK2 - STACK1$
Multiplication: $STACK1 = STACK2 *n STACK1$
Division: $STACK1 = STACK2 / STACK1$

Note: On the multiply operator, the n parameter specifies the descaling factor to apply to the result. (It doesn't work, but is supposed to).

➡ Concatenation Operator

: (colon)
 $STACK1 = STACK2 : STACK1$

➡ Relational Operators & Effects⁵

= Equal to
If $STACK2 = STACK1$ then $STACK1 = 1$ else $STACK1 = 0$

Not equal
If $STACK2 \# STACK1$ then $STACK1 = 1$ else $STACK1 = 0$

⁵ Note that following conditional functions are backwards between generic Pick and the SMA standard.

< Less than

If $STACK1 < STACK2$ then $STACK1 = 1$ else $STACK1 = 0$

> Greater than

If $STACK1 > STACK2$ then $STACK1 = 1$ else $STACK1 = 0$

[Less than or equal to

If $STACK1 \leq STACK2$ then $STACK1 = 1$ else $STACK1 = 0$

] Greater than or equal to

If $STACK1 \geq STACK2$ then $STACK1 = 1$ else $STACK1 = 0$

➡ **Functional Operators**

R Remainder

$STACK1 = R(STACK2/STACK1)$

S Sum

$STACK1 = \text{Summation of multivalues into } STACK1$

P Duplicate

Pushes a copy of $STACK2$ into $STACK1$. All stack positions push one position.

— Exchange

Exchanges the values of $STACK1$ and $STACK2$.

[] Extract

$STACK1 = STACK3[STACK2,STACK1]$ where $STACK3$ contains the string, $STACK2$ has the starting position, and $STACK1$ has the length. Result goes in $STACK1$.

: Concatenation

$STACK1 = STACK2 : STACK1$.

? Mystery Operator

This is actually part of the code. We don't know, however, what it does.

(conversion)

Any ACCESS conversion may be used as an operator. The top entry of the stack is used as the source value for the conversion, which must be enclosed in parentheses.

➡➡ Special Operands

LPV

Loads result of last conversion on top of stack.

NA

Loads the number of attributes contained in the item on top of the stack.

Compatibility: SMA

NB

Number of break level.

ND

Returns the number of detail lines. Used only on line 7 (conversion).

NI

Returns the number of the item being processed.

NL

Loads the length of item on top of stack.

Compatibility: SMA

NV

Returns the number of the multi-value being processed.

NS

Returns the number of the Sub-value being processed.

FS{*n*};*element*{*element* . . . }

Compatibility: SMA

The SMA standard form of the F correlative differs from the generic Pick form. First, the letter “S” follows the F to indicate that this is the “standard” form. The other difference between the SMA standard and the generic Pick version is in the handling of relational operators:

< Less than

If $STACK2 < STACK1$ then $STACK1 = 1$ else $STACK1 = 0$

> Greater than

If $STACK2 > STACK1$ then $STACK1 = 1$ else $STACK1 = 0$

[Less than or equal to

If STACK2 <= STACK1 then STACK1 = 1 else STACK1 = 0

] Greater than or equal to

If STACK2 >= STACK1 then STACK1 = 1 else STACK1 = 0

Note that the references to STACK1 and STACK2 reversed. It's worth the time to test this function on your system with tests of the conditionals. For instance:

F;C2;C1;<.

G{*skipsegments*} *delimiter* *getsegments*}

Group extract. Retrieves data elements from a string containing user specified (non-reserved) delimiters. Note that the implied spaces between the arguments in this correlative are *not* part of the syntax, thus, they are not allowed. They were put there just to provide visual relief for the reader. See the examples below for actual samples.

The *skipsegments* argument indicates the number of value segments to skip over. It must be numeric. If omitted, zero is used.

The *delimiter* may be any single non-reserved and non-numeric character that delimits part of a stored value. The reserved characters are attribute, value, subvalue and segment marks.

The *getsegments* argument determines the number of groups to retrieve from the data value. Here are some examples:

Stored data string = CA*92714*1000

Conversion	Result
G*1	CA
G0*1	CA
G1*1	92714
G2*1	1000
G1*2	92714*1000

L{*maxlength*{*maxvalue*}}

L0

Length function. Specifies output length restrictions on data fields. The *maxlength* parameter indicates that the value will be output only if the length of the string is less than the *maxlength* parameter, otherwise a null is returned. The L command, followed by a 0 (zero), returns the length of the data field. Here are some examples:

- L0 Returns the length of field
- L9 Outputs data if field length is less than nine characters
- L3,9 Outputs data if field length is greater than three characters and less than nine characters

M(just){(precision){scalefactor}}(Z){,} {signcode}(\$) {formatmask}

Mask (Decimal) function. Performs output conversions, literal insertions and currency formatting on an input converted attribute. The justification specified in the *just* parameter may be either an L (left) or an R (right). The *precision* parameter specifies the number of decimal positions to print after the decimal point. It must be between 0 (zero) and 9.

The scaling factor specified in the *scalefactor* parameter indicates the power to which the value is to be descaled. It must be between 0 (zero) and 9. The *Z* parameter suppresses zero balance fields. The , (comma) inserts commas in the thousands and millions position on output.

The *signcode* parameter alters the normal handling of negative numbers, with the exception of the “D” signcode. Their functions are as follows:

- C Negative values are followed by the literal, “CR”
- D Positive values are followed by the literal “DB”
- E Negative values are enclosed in angle brackets: <value>
- M Negative numbers are followed by a – (minus) sign
- N Suppresses (leading) minus sign on negative numbers.

The \$ parameter appends a “dollar sign” to the value prior to justification.

The *format mask* may be any combination of literals and format operators:

- #n Fills with “n” blanks
- *n Fills with “n” asterisks
- %n Fills with “n” zeros

Here are some examples:

<i>Internal Data</i>	<i>Conversion</i>	<i>External Data</i>
12345	MR2	123.45
123456	MR2,\$	\$1,234.56
123456	MR4,\$	\$12.3456
– 123456	MR2,\$	\$1,234.56
– 123456	MR2,E\$	<\$1,234.56>
– 123456	MR2,C\$	\$1,234.56CR
123456	MR2,\$*12	\$***1,234.56

MC{/}code

Mask character function. Performs various string conversions, such as upper-to lower case and vice versa. The following is a list of Character Conversion Codes:

MCA	Retrieves all alphabetic characters from data value
MC/A	Retrieves all non-alphabetic characters and ignores alphabetic characters
MCDX	Converts decimal data value to its hexadecimal equivalent
MCL	Converts all upper case characters to lower case and ignores lower case characters
MCN	Retrieves all numeric characters from data value
MC/N	Retrieves all non-numeric characters from data value
MCP	Converts all non-printable characters to periods. Non printable characters are those between the hexadecimal values X'00'—X'1F' and those between X'7F'—X'FB'. Note that the characters above X'7F' display, but actually have different character meanings as their high order bit is lost.

Compatibility: Pick⁶

MCT	Converts uppercase characters to lowercase, starting with the second character in each word. (Capitalizes the first character of each word after a non-alphabetic character).
MCU	Converts all lower case characters to upper case and ignores upper case characters
MCXD	Converts hexadecimal data value to its decimal equivalent

MS

Mask sequence function. Used to sort an attribute in a sequence specified by the item called "SEQ" in the ERRMSG file. Only used on an attribute defined as a sort key, thus, this code must be applied as a correlative (attribute 8). Not available for use in PICK/BASIC. The SEQ item must be manually added to the ERRMSG file. This is the suggested format for the item:

```
SEQ (Item-id)
001 AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUu . .
. . VvWwXxYyZz 01234567890 ,?!'";:~* / ^ = . .
. . ) [ ( ) < > @ # $ % & ' ` \ :
```

Compatibility: Pick 2.2 and higher

⁶ Note that this still works on most implementations other than Microdata and Ultimate.

MT{(H){S}}

Mask time conversion. Converts internal time to a formatted external output.

- H Specifies a 12 hour clock. Default is (military) 24 hour clock.
- S Includes seconds in output value.

Here are some examples of Time Conversions:

<i>Internal Data</i>	<i>Conversion</i>	<i>External Data</i>
3600	MT	01:00
46800	MT	13:00
3600	MTH	01:00AM
46800	MTH	01:00PM
3630	MTS	01:00:30
46800	MTHS	01:00:00PM

MX

Mask hexadecimal function. Converts any decimal or ASCII character string to its hexadecimal equivalent.

MY

Mask hexadecimal compression. Converts a hexadecimal string, two bytes at a time, into the corresponding ASCII one-byte representation.

Compatibility: SMA

P(matchstring){;(matchstring) . . .}

Pattern matching function. The matchstring may be a composite of literals and/or match operators, appended to length specifications. To be accepted for processing, the value must be the exact length of the length parameter. The following are match operators:

- nA “n” alphabetic characters only
- nN “n” numeric characters only
- nX “n” alphanumeric characters
- “text” Any quoted text string

A length of 0 (zero) allows any length of the following match operator. Multiple *matchstring* parameters will accept the data if any of the match-strings are met.

R*minrange,maxrange*{*minrange,maxrange* . . . }

Range function. Outputs data field only if data value falls between the ranges specified in *minrange* (minimum) and *maxrange* (maximum) parameters.

Multiple range specifications allow display of the data value if the data value falls between any of the range pair sets. In using multiple range set specifications on negative values, the ranges should be in ascending sequence, beginning with the lowest.

S;*attrnum;attrnum*

S;*'text';'text'*

S;**;'text'*

Substitution function. Substitutes data value from the attribute referenced in *attrnum* if current data value is null or zero, else the string specified in *text* is displayed.

The *** is used to display the original data value, if it is not null or zero. Otherwise, the second argument is displayed. This feature may be embedded within an F-correlative in the following form:

Fattrnum(S;**;'text'*)

It also may be embedded within an A- correlative.

T{*startcolumn*,}*length*

T{*startcolumn*}

Text extraction (substring) function. Retrieves any number of characters from an attribute, starting at the position specified in *startcolumn*, for a length of *length*. Both parameters must be numeric.

If the *length* parameter is not specified, the extraction begins from the same end of the string as the justification attribute (9), either L (left) or R (right).

T{**DICT** }*filename;code{vmc};inamc;outamc{;breakamc}*

File translation function. Allows an attribute value to translate through another file. The attribute referenced in line 2 of the dictionary definition item is assumed to be an item-id in the specified filename. To retrieve a

specific value within the attribute, the optional value mark count may be specified in the *vmc* parameter.

The translate code determines the action to take with unsuccessful translate values. Here are the translate codes:

- C Displays original value, if value does not exist in translate file
- Cn Functions like the “C” code. The “n” represents a value mark count of the value to retrieve
- I Input verify only. Same functions as V code on input and C code on output
- O Output verify only. Same functions as C code on input and V code on output
- V Conversion item must exist and the specified attribute must have a value, otherwise an error message will display
- X Displays a null, if value does not exist on translate file

The *inamc* parameter specifies the Attribute Mark Count to use for input conversion. The default is null. The *outamc* parameter specifies the Attribute Mark Count of the attribute in the translate file that contains the value(s) to output. The Attribute Mark Count referenced in the *breakamc* specifies to use this attribute for the BREAK-ON and TOTAL lines, rather than the “outamc”.

USER EXITS AS CORRELATIVES OR CONVERSIONS

User exits are special functions that make direct references to assembler routines known as *modes*. They were added to the system to fulfill needs outside the standard boundaries of the PICK System prior to being replaced by extensions to the system. User exits have always been dangerous, and as long as they remain, will continue to be. *EXTREME* caution is advised in using them. Caveat Emptor.

Note that the topic of user exits was omitted from the SMA standards. The general form of user-exits in an Attribute-Defining Item is:

```
item-id
001 pointer type
002 AMC
003 column heading
004
005
006
007 {user exit}
008 {user exit}
009 justification
010 column width
```

User exits are generally interchangeable with the standard conversions and correlatives. This is the Structure of User Exits:

Uefff

Where *e* is the (decimal) entry point, and *fff* is the (hexadecimal) frame-id.

USER EXITS

The following is a table of User Exits:

<i>User Exit</i>	<i>Description</i>
U201E	Returns the number of attributes in the current item
U0070	Performs correlated match on multi-valued set using the format: U0070,D1,D2 where D1 is match and D2 is result to display
U1070	Returns checksum value for current item
U2070	Returns checksum value for specified frame
U3070	Returns PCB-FID as four hexadecimal characters
U4070	{Re-)enable output (print on)
U5070	Converts string abcdef to the form abc-000. If string is less than 3 characters, only the string is returned
U6070	Input up to 100 characters. A <cr> is output before and after the input. The prompt character is used
U7070	Input up to 100 characters. The prompt character is used, and there is no other output
U9070	Toggles the page format flag (NOPAGE)
UA070	Returns the specified value from a multi-valued set. The value number is obtained from the valuestring portion of the selection criteria in the ACCESS sentence
UC070;n	Returns value number "n" from multi-valued set
U508E	Activates the COPY processor from ACCESS. The HEADING and FOOTING modifiers are subsequently available. See also the LIST-ITEM and SORT-ITEM commands which effectively do the same thing
V;;n	Code used by the WITHIN connective and placed in attribute 8 of the D-pointer to the DATA section of the file. See the WITHIN connective for further information

Proc



Please refer to the glossary at the back of the Pocket Guide. It contains definitions of the standard terms used in the PICK system, along with the special terms that were created during the production of this guide.

➤ About SMA Standards

Everything in this section except for the (— n) functions of the T command is compatible with the SMA standard document 501, dated March 1987.

ABOUT LOGON PROCS

A *logon PROC* is one that starts automatically upon logging on to the system. This is how most systems whisk the user away into a menu when they log on. The way it works is that after the user has provided a valid account name and optionally the password, the MD of the account is checked to see if there is a PROC with the same name as the account. If such a PROC exists, it starts up automatically. Obviously, if no such PROC exists, control is transferred to TCL. While logon PROCs are optional, they are useful for issuing several TCL commands, such as TERM-TYPE and SET-SYM.

PROC LINKAGES

{{DICT} *filename* {*itemname*}}
{DICT} *filename* {*itemname*}) {*label*}

Transfers control to PROC in specified *filename*. If no *itemname* is specified, the default *itemname* is the value in the current position of the input buffer. Control is not returned to the calling PROC. The *label* parameter indicates the statement label number where execution should begin.

[{DICT} {*filename*} {*itemname*}]
[{DICT} *filename* {*itemname*}] {*label*}

As above with the former linkage, but control returns to calling PROC upon completion. If no *itemname* is specified, the *itemname* defaults to the value in the current position of the input buffer. Here are some examples:

[PROCS EXAMPLE1]

Transfers control to EXAMPLE1 in the file called PROCS. Control returns to the next line upon completion.

[PROCS EXAMPLE2] 25

Transfers control to EXAMPLE2 in the file called PROCS, then begins execution at statement label 25. Control returns to the next line upon completion.

[] 150

Transfers control to statement label 150 in the current PROC and begins execution.

(PROCS)

Transfers control to the PROCS file, using the item-id indicated in the input buffer.

PROC COMMANDS

+ number

Adds value of decimal number to the current parameter of the presently active input buffer.

— number

Subtracts value of decimal number from the current parameter of the presently active input buffer.

A{*surroundchar*}{(*parnum*){,*charcount*()}}

Moves data from the specified parameter number to the currently active output buffer. The *surroundchar* parameter specifies the character to place before and after the string, after it is moved. The *charcount* parameter specifies the number of characters to move. Here are some examples

A"1

Moves contents of input buffer one to the currently active output buffer, surrounded by double quotes.

A(2,999)

Moves the second through 999th (or whenever it runs out of characters) to the currently active output buffer.

B

Decrements input pointer by one.

BO

Decrements output pointer by one.

C {*text*}

Comment. All data on line ignored by PROC processor.

D{(*parnum*){,*numbercharacters*()}}{+}

Displays contents of the current parameter in the currently active input buffer. The Default is the currently active input buffer. If the optional *parnum* (parameter number) is specified, only the contents of the specified input buffer are displayed. The D command, followed by a 0, displays the entire current input buffer. The + character suppresses the automatic linefeed at the end of the display.

F

Increments input pointer by one.

GO *statementlabel*

G *statementlabel*

GO A{*parnum*}

Transfers control to specified statement label. The first two forms transfer control directly, meaning that the statement label is explicitly stated. The latter form allows an “indirect” transfer according to the value of the label as extracted from a parameter in the input buffer.

Hstring{<}

Moves specified text string to currently active output buffer. The < character is the equivalent of a carriage return, and must follow each parameter when moving multiple parameters to the secondary output buffer (following the STON command).

IF {#} E {*operator* ERRMSG*number*} *command*

Tests for specified ERRMSG number. If true, performs specified PROC command. For example:

IF E = 401 XNo items were selected . . .

IF {#} S *command*

Tests for active list. may be used after a SELECT, SSELECT, QSELECT or GET-LIST command.

IF {#} A{*parnum*}{,*n*} {*operator* {*matchstr*}} *command*

Tests contents of buffer specified in A{*parnum*} expression with any logical or pattern matching operator, and performs the specified PROC command if successful. The optional *n* argument allows only the specified number of characters.

Logical Operators

=	Equal to
#	Not equal to
<	Less than
>	Greater than
[Less than or equal to
]	Greater than or equal to

The # character tests for the absence of the expression, when used on the left of the A command.

Pattern Matching Operators

- (nA) Accepts *n* alphabetic characters only
- (nN) Accepts *n* numeric characters only
- (nX) Accepts *n* of any characters
- ('string') Accepts literal string

(Note that the pattern matchstrings must be enclosed in parentheses)

The *n* parameter in the pattern match string specifies the length of the match operator field. A length specification of zero (0) allows variable length input. Here are some examples:

```
IF A # (3N'-'2N'-'4N) OInvalid SS#.
IF A = X XDone.
IF A = (0N) GO 10
IF A1 = QUIT X PROC Terminated Voluntarily
IF A1,1 = Q X PROC Terminated Voluntarily
IF # A1 GO 10
IF A1 = (0N) GO A1
```

**IH **

Removes the current parameter from the currently active input buffer and replaces it with a null value. If the pointer is in the middle of a parameter, the parameter is truncated, starting at the current position.

**IH **

Inserts a new null parameter in front of the current parameter in the currently active input buffer. If the pointer is in the middle of a parameter, the parameter is truncated starting at the current position and a new null parameter is added after the truncation.

IH *string*

Inputs data string into active input buffer.

IP(*promptcharacter*)

Inputs data from terminal into active input buffer. The default prompt character is a colon (:).

IS(*promptcharacter*)

Inputs data into secondary input buffer. The default prompt character is a colon (:).

IT

Inputs data from tape label into primary input buffer. As of release 2.2, two responses are now available to the PROC "IT" (Input-from-Tape) command. They are **C** to continue or **O** to override character.

O {*text*}{+}

Outputs literal text to terminal at next available line on screen. The + character suppresses the automatic linefeed/carriage return after output.

P

Processes command currently in output buffer.

PH

Processes command currently in output buffer, and suppresses ("hushes") all terminal output from PROC command.

PP

Prints output buffers prior to processing.

PW

Prints output buffer prior to processing, prompts with a question mark, and waits for user response before execution. Here are valid responses:

G (or <cr>)	Go. Proceed with process
S	Skip. Proceeds with next command in PROC
X (or N)	Cancel. Returns control to TCL

PX

Processes command in output buffer, stops execution of PROC upon completion.

Combining Commands

Several of the commands already defined may be combined, as shown in the following examples:

PHX

Process, hush output, terminate.

PWH

Print, wait, process and hush.

PQ

Required on first line of every PROC.

RI(*startparameter*)

Resets both input buffers to null. Use of *startparameter* resets to null all parameters following specified parameter and the entire secondary input buffer.

RO

Resets both output buffers to null.

S(*parnum*)

“Sets” the input pointer to the specified parameter number; activates specified parameter number as currently active input buffer.

SP

Selects the primary input buffer, and positions the pointer to the beginning of the input buffer.

SS

Selects the secondary input buffer, and positions the pointer to the beginning of the secondary input buffer.

STOFF

ST OFF

Selects primary output buffer and directs subsequent output to primary output buffer.

STON

ST ON

Selects secondary output buffer and directs subsequent output to secondary output buffer.

T {*function,function, . . .*}{+}

Directs output to terminal and controls special terminal display functions: The following table list the functions of the T Command:

<i>Function</i>	<i>Description</i>
+	Suppresses carriage return after output
B	Rings terminal "bell"
C	Clears terminal screen
(column,row)	Outputs value at column and row coordinates
In	Outputs ASCII value of the integer character indicated by "n"
Xx	Outputs HEX/ASCII value of the hexadecimal character indicated by "x"
"text"	Outputs literal text

Compatibility: Pick SMA

Special Control Functions:

- @(-1) Clears the screen and positions the cursor to home position (0,0).
- @(-2) Positions the cursor to the home position (0,0)
- @(-3) Clears to the end of screen from the current cursor position
- @(-4) Clears to the end of the line from the current cursor position

Compatibility of above: Pick SMA

The following are compatible only with generic Pick and the AT/XT releases:

- @(-5) Enables the blink function
- @(-6) Disables the blink function
- @(-7) Enables the protect function
- @(-8) Disables the protect function
- @(-9) Moves the cursor position back one position
- @(-10) Moves the cursor up one line

The following additional control functions have been added to the T command on the Pick XT and AT releases:

- (-11) Activates screen protect
- (-12) Deactivates screen protect
- (-13) Activates reverse video
- (-14) Deactivates reverse video
- (-15) Activates underlining
- (-16) Deactivates underlining

The following control functions may only be used on Port 0 (zero), equipped with a color graphics display on the Pick XT or AT:

Background Color Controls:

- (− 33) White
- (− 34) Brown
- (− 35) Magenta
- (− 36) Red
- (− 37) Cyan
- (− 38) Green
- (− 39) Blue
- (− 40) Black

Foreground Color Controls (Full Intensity):

- (− 41) White
- (− 42) Brown
- (− 43) Magenta
- (− 44) Red
- (− 45) Cyan
- (− 46) Green
- (− 47) Blue
- (− 48) Black

Foreground Color Controls (Half Intensity):

- (− 57) White
- (− 58) Brown
- (− 59) Magenta
- (− 60) Red
- (− 61) Cyan
- (− 62) Green
- (− 63) Blue
- (− 64) Black

Additional Control Functions:

- (− 89) Activates monochrome display unit in 80 by 25 black and white mode
- (− 93) Activates color graphics monitor in 80 by 25 color mode
- (− 94) Activates color graphics monitor in 80 by 25 black and white mode

Uentrypoint modeid

Transfers control to user assembly code subroutine. Must be 4-digit hexadecimal number. Note that the following “infact” user modes have unofficially been part of the Pick system for many years and since they are not “official”, no support is usually provided for them. Note also that if you are

not extremely careful in using these, you can easily crash your machine and possibly do harm to your data. We assume no liability for errors and/or omissions, and have recorded these for posterity's sake. Caveat Emptor. The following User-Exits are available in the PROC language:

<i>Code</i>	<i>Function</i>	<i>Format</i>
U01A6	Used for cursor control. Now made obsolete by the "T" command. <i>Control Codes:</i> (col,row) B C In Xx "text"	U01A6 control code{,control code . . .} (continuation of PROC) Positions cursor at specified column and row Rings bell Clears screen Outputs integer value of "n" Outputs hexadecimal character "x" Outputs literal text.
U01AD	Retrieves a value from an attribute of an item in a file. <i>Output-Codes:</i> A P S T V VA <i>Note:</i> this function allows for indirect references to values from the input and output buffers by using the following special sym- bols: % # Reads attribute 1 of the item called CO.NAME in the file called CON- TROL-FILE and places it in the current output buffer	U01AD filename itemname amcexp output-code error-return line (continuation of PROC) Outputs to alternate output buffer Outputs to primary input buffer Outputs to current output buffer Outputs to terminal Verifies that file exists; No output Verifies non-null attribute Current input buffer value Current output buffer value <i>Direct reference example:</i> PQ U01AD CONTROL-FILE CO.NAME 1 S XCan't find CO.NAME or CONTROL-FILE

<i>Code</i>	<i>Function</i>	<i>Format</i>
	Reads the attribute number as defined in the fourth location of the input buffer, using the item-id from the third location and the filename from the second location.	<i>Indirect reference example:</i> PQ U01AD %2 %3 %4 P XSomething went wrong . . . D0
U11AD	Retrieves multiple values from an attribute of an item in a file. See U01AD for format	
U01B8	Retrieves and optionally formats a value from a file or a text string	
U01BC	“n”-way branch	
U11BC	Pads value in secondary input buffer with zeroes (0's)	U11BC number-of-zeroes (continuation of PROC)
U218D	Disables break key	
U21A2	Functions exactly like the RO command, resetting the Primary (and Secondary) output buffers	
U21AD	Retrieves multiple values from an attribute of an item in a file, without advancing multi-value counter (See U01AD for format)	
U21BC	Deletes all entries, or last entry, from output buffer. When <i>condition-flag</i> returns a zero (0), all entries are deleted. When one (1) is returned, just the last entry is deleted.	U21BC condition-flag

<i>Code</i>	<i>Function</i>	<i>Format</i>
U307A	Functions exactly like the (TCL) command, "SLEEP untiltime", instructing the process to sleep until the specified military time.	
U318D	Enables break key	
U31AD	Returns current port number See U01AD for explanation of output codes.	U31AD output-code (continuation of PROC)
U31BC	Subroutine call	
U407A	Functions exactly like the (TCL)command, "SLEEP numberseconds", instructing the process to sleep for the specified number of seconds.	
U41AD	Replaces string in the primary input buffer	U41AD string (continuation of PROC)
U41BC	Subroutine recall	
U50BB	Functions exactly like the (TCL) command, "WHO", returning the current port number and account name	
U51BC	Subroutine return	
U61BC	Functions exactly like the "PH" command, disabling terminal output from the process	
P91BC	Transfers to another PROC, without affecting current input buffer(s)	Hprocname P91BC

<i>Code</i>	<i>Function</i>	<i>Format</i>
	Note that this user-exit must follow the letter p (to Process).	
UA1BC	Moves entries from the current pointer position of the input buffer to the end of the input buffer	
UE070	Returns the current reel number to the active input buffer (Added in release 2.2)	
UD070	Returns “WHO” output to active input buffer	

X {*text*}

Terminates PROC and outputs text on terminal. Also used to designate the end of an internal and external PROC subroutines.

Runoff



Most of RUNOFF is compatible with the SMA standard. The exceptions to the SMA standard are noted and occur only in the options available with the HEADING and FOOTING commands. Only one command in generic Pick—the .NOPAGING command—is not part of the SMA standard.

FORMAT OF THE RUNOFF COMMAND

>RUNOFF {DICT} *ilename itemlist* {(options)}

Activates output function of RUNOFF text processor. The options are:

- n* Any integer number. Specifies the number of times to overstrike a character or characters designated as boldface
- C Suppresses .CHAIN and .READ output
- I Outputs next item name
- J Suppresses highlighting function
- N Activates NOPAGE function on output to terminal
- P Directs output to system printer, via spooler
- S Suppresses boldface and underline function
- U Outputs all characters in upper case

SPECIAL CONTROL CHARACTERS

@	Prints following character in boldface
@^	Activates boldface function
@\	Deactivates boldface function
&	Prints following character underlined
&^	Activates underline function
&\	Deactivates underline function
^	Prints next character in upper case
^^	Activates upper case function
\	Prints next character in lower case
\\	Deactivates upper case function
—	Treats the following character as a literal, when using reserved RUNOFF characters as text items
<	Begins following text at next tab stop
>	Ends following text at next tab stop

RUNOFF COMMANDS

Note that many RUNOFF commands have an acceptable abbreviated form. The abbreviated form follows all of the full commands that recognize the abbreviated form.

. {text}***

Comment line. All text or commands on line are ignored by RUNOFF processor.

.BEGIN PAGE

.BP

Terminates current page, prints optional FOOTING, ejects a page, increments page counter, and prints optional HEADING.

.BOX *leftmargin, rightmargin*

.BOX OFF

Encloses following text in a “box” of vertical bars on the output page. The BOX command, followed by the OFF parameter, disables the .BOX function.

.BREAK

.B

Outputs partially filled line before processing next line.

.CAPITALIZE SENTENCES

.CS

Activates automatic capitalization of sentences following a period, exclamation point, question mark, colon, or a semicolon that is followed by one or more spaces.

.CENTER

.C

Centers following line between current left and right margins.

.CHAIN {DICT} {filename} itemname

Transfers control to specified *itemname*. If a file name is not specified, the current file name is used. Control does not return to the source item. See also the .READ command.

.CHAPTER text

Ejects page, increments page and chapter counters and outputs the literal, “CHAPTER”, followed by the optional text centered between the current margins. The chapter, page number, and text are accumulated for the table of contents.

.CONTENTS

Outputs the current table of contents, as defined by previous .CHAPTER and .SECTION commands.

.CRT

Directs output to terminal.

.EC

End case. Disables any previous .UPPER CASE or .LOWER CASE commands.

.FILL

.F

Activates line fill function. Words from the source item are printed one at a time on the output line until no more words can fit on the line, and then a new line is begun. In other words, individual lines in the source item are combined to form continuous paragraphs of the proper width. See also the .STANDARD command.

.FOOTING

{{text} {options} . . .}

Defines following line as text to output at the bottom of each page. If the line following the .FOOTING command is null, then the footing is suppressed. The following is a list of options:

- C** Centers line on page according to current page width as set by the TERM command
- D** Outputs current system date
- F** Outputs filename
- Fn** Outputs filename left-justified in “n” spaces. (Not part of SMA standard)
- I** Outputs item-id
- In** Outputs item-id left-justified in “n” spaces. (Not part of SMA standard)
- L** Performs carriage return/linefeed
- P** Outputs current page number, right-justified in a field of four spaces
- Pn** Outputs current page number, left-justified in “n” spaces. (Not part of SMA standard)
- PN** Outputs current page number, left-justified (Not part of SMA standard)
- T** Outputs current system time and date

.HEADING

{{*text*} {'options'} . . .}

Defines following line as text to output at the top of each page. If the line following the .HEADING command is null, then the heading is suppressed. The following are available options:

- C Centers line on page according to current page width as set by the TERM command
- D Outputs current system date
- F Outputs filename
- Fn Outputs filename left-justified in *n* spaces. (Not part of SMA standard)
- I Outputs item-id
- In Outputs item-id left-justified in *n* spaces. (Not part of SMA standard)
- L Performs carriage return/linefeed
- P Outputs current page number, right-justified in a field of four spaces
- Pn Outputs current page number, left-justified in *n* spaces (Not part of SMA standard)
- PN Outputs current page number, left-justified (Not part of SMA standard)
- T Outputs current system time and date.

.HILITE {*character*}

.HILITE OFF

Prints the optionally-specified character along the right margin. If the character is omitted, or the word “OFF” appears after the command, then the HILITE function is disabled. Any character, other than a period (.), may be specified as the highlight character.

.INDENT {-}*numberspaces*

.I {-}*numberspaces*

Indents the following line the specified number of spaces. Left margin is not reset. The *numberspaces* may contain a negative number to cause “out-denting”. If the *numberspaces* argument is omitted, one (1) is assumed.

.INDENT MARGIN {-}*numberspaces*

.IM {-}*numberspaces*

Increments the left margin by the specified number of spaces. The *numberspaces* may contain a negative number to move the left margin farther to the left.

.INDEX text

Inserts text with current page number into index. Text must be enclosed in double quotes if it contains embedded blanks. See also the .PRINT INDEX command.

.INPUT

Prompts for input from terminal screen. The entered text is then output as though it had been stored in the document.

.JUSTIFY

.J

Activates fill mode and justifies lines to the right margin by placing extra spaces between words on the line. See also the .STANDARD and the .FILL commands.

.LEFT MARGIN *marginsetting*

Sets left margin tabstop to specified column number. Left margin is automatically set to zero (0) in the .STANDARD command.

.LINE LENGTH *numbercharacters*

Specifies the maximum number of characters per output line. This number, plus the left margin setting, (minus one), denotes the right margin position. Line length is automatically set to seventy (70) in the .STANDARD command.

.LOWER CASE

.LC

Outputs all text in lower case. See also the .EC command.

.LPTR

Directs output to system printer, via spooler.

.NOCAPITALIZE SENTENCES

.NCS

Deactivates automatic capitalization of sentences function.

.NOFILL

.NF

Disables line fill and justification functions. Outputs text line for line exactly as stored. See also the .FILL and .JUSTIFY commands.

.NOJUSTIFY

.NJ

Disables justification function. Line fill function remains in effect and lines are output with a “ragged” right margin. See also the .JUSTIFY command.

.NOPAGING

.N

Activates NOPAGE function on output to terminal. (Not part of SMA standard).

.NOPARAGRAPH

Disables paragraph function. See also the .PARAGRAPH command.

.PAGE NUMBER *pagenumber*

Sets current page number to specified number. If “pagenumber” is omitted, one (1) is assumed.

.PAPER LENGTH *paperlength*

Specifies the maximum number of print lines per output page.

.PARAGRAPH {-}*numberspaces*

.P {-}*numberspaces*

Specifies number of spaces for paragraph indentation. Any line beginning with a blank is indented the specified number of spaces. The *numberspaces* may contain a negative number to cause the paragraph to “outdent”.

.PFILE *printfile#*

Directs subsequent output to specified *printfile* number, just like the PRINT ON statement in BASIC. Typically used to provide a separate spooler entry for containing the table of contents created with the .CONTENTS command.

.PRINT text

Displays following line of text on terminal. Typically used to provide a prompt message for use with a subsequent .INPUT command.

.PRINT INDEX

Outputs index created with previous .INDEX commands. Note that this may not be used with the .SAVE INDEX command. See also the .INDEX command.

.READ {DICT} {*filename*} *itemname*

Transfers control to specified RUNOFF item for output and returns control upon completion. If the filename is omitted, the current filename is used. See also the .CHAIN command and the I option with the RUNOFF command.

.READNEXT

Extracts top entry of list created by an external {S}SELECT statement prior to output and inserts next value into text of item. See also the {S}SELECT command in RECALL section.

.SAVE INDEX *filename*

Saves entries defined in previous .INDEX commands in specified filename, using the item-id of the current RUNOFF item.

The *filename* of the index must be different than the RUNOFF text *filename* to avoid DESTROYING data.

.SECTION *section-number text*

Specifies section level number to increment by one and stores the resulting section number and optional text in the table of contents. See also the .CONTENTS command.

.SET TABS *tabstop*{*tabstop* . . .}

Specifies tabstop locations at specified column position(s). *Tabstops* must be in ascending sequence, and zero (0) is not valid. Up to 30 tabstops may be set. See also the C (column) command in the EDITOR section.

Any word following the “less-than” symbol (<) will *begin* at the next tab stop. Any word following the “greater-than” symbol (>) will *end* at the next tab stop. Note that tabs are only in effect while in .NOFILL mode.

.SKIP {*numberlines*}

.SK {*numberlines*}

Outputs specified number of blank lines on page, taking into account any previously specified .SPACING commands. If “*numberlines*” is omitted, then one (1) is assumed.

.SPACE *numberlines*

.SP *numberlines*

Outputs specified number of blank lines on page, regardless of any previously specified .SPACING commands. If *numberlines* is omitted, then one (1) is assumed.

.SPACING *numberlines*

Specifies inter-line spacing to the specified number of lines (e.g. .SPACING 2 sets double spacing). See also the .SK and .SP commands.

.STANDARD

Specifies for RUNOFF to use the following default parameters:

.CAPITALIZE SENTENCES

.CRT

.FILL

.FOOTING (null)

.HEADING (null)

.JUSTIFY

.LEFT MARGIN 0

.LINE LENGTH 70

.PARAGRAPH 5

.UPPER CASE

The SMA standard for .STANDARD is:

.CAPITALIZE SENTENCES
.FILL
.JUSTIFY
.FOOTING (null)
.HEADING (null)
.LEFT MARGIN 0
.LINE LENGTH 70
.PARAGRAPH 5
.END CASE

.TEST PAGE *numberlines*

.TP *numberlines*


Counts the number of lines remaining to output on the current page. If the *numberlines* parameter is less than the count returned, then the following text is output on the current page, otherwise, a page is ejected and the text is output on the new page. This prevents blocks of text from being split across a page boundary.

.UPPER CASE

.UC

Outputs text in uppercase characters, unless specifically altered by the lowercase special control character, (\). See also the .EC command.

Jet



This chapter covers release 2.0 of JET “The Works” word processing software.

TCL COMMANDS

>JET-EDIT *filename itemlist**

Activates JET full-screen editor mode.

>JET-IN *filename itemlist* ((options))*

Activates word processor. Here are the available options:

- B Positions cursor to bottom of document
- N Activate paragraph numbers
- R Reveal underlining & boldfacing
- V Disables view mode

>JET-OUT *filename itemlist* ((options))*

Produces output of items previously entered via word processor. The options are:

- A Anchor. Suppresses a form-feed at the beginning of any document and will suppress the initialization of the line counter
- B Auto Bin. Selects bin 1 for the first page and bin 2 for remaining pages
- C Directs output to console printer. Sometimes known as a slave printer
- F Formfeed. Forces new page between each item
- H Hard-copy. Designates a printing terminal
- J Printer Driver. For printers that use table-driven printer drivers (such as the HP LaserJet) to activate letter quality features
- L Line#. Prints paragraph line numbers
- M Manual Feed. Causes a pause of each page, after the form feed ejects the sheet, and before the next page is fed into the printer
- N Nopage. Prints without breaking at end of each page
- P Print Spooler. Sends the print job to the SPOOLER for final output on a serial or line printer
- Q Quality. Indicates a letter-quality printer
- S Micro-Justify. Available on letter-quality printers and refers to 'micro-justification', instead of random blank spacing, to justify final copy
- T File Writes an item directly to a file
- V View. Prints the ruler and backslash command lines on the hard copy
- Z Line Numbers. Prints consecutive print line numbers to the left margin. Numbers repeat on each succeeding page

KEY FUNCTIONS

Release 2.0 has two modes of operation; *Function key operation* allows assigning the JET commands to function keys; *Alpha key operation* uses the keyboard keys for the JET commands. The following describes alpha key operation.

MOVING THE CURSOR

- T Top of document
- B Bottom of document
- G Go to paragraph#
- 0 Previous page
- 1 Previous sentence
- 2 Down one line
- 3 Next sentence
- 4 Previous character
- 5 Top of page
- 6 Next character
- 7 Previous word
- 8 Up one line

9 Next word
. Next page
, Go to page
^ End of line
{ Previous paragraph
} Next paragraph

SPECIAL KEY COMMANDS

J Special key

Options are:

C(apture)
E(xecute)
S(ort columns)
U(Spell check on input)
W(indows)

INSERT MODE

I Insert text

L Insert line

<cr> Insert blank line

or

<New Line>

U Spell check on Input

<ctl>A Insert “hard blank”

REPLACING TEXT

W Replace word
R Replace text
@ Replace(mode)

EDIT MODE

A	Another Search	Z	Transpose chars.
C	Clear screen	!	Change case, line
E	Edit ruler	=	Paginate
F	File/Exit	/	Display page/line
H	Undelete]	Start/End cut
M	Merge	N	Para.#s on/off
N	Para.#s on/off	:	Change case, char.
O	Overview	-	Hyphenate
P	Paste	+	Spell check
S	Search/replace	?	Help
V	View/Overview	#	Quit cut
;	Reverse Search		

RULER EDIT COMMANDS

E	Edit Ruler Mode		
A	Set auto tab	S	Save ruler S
B	Set L and R	X	Exit (don't embed) X
C	Set center tab	Z	Set hyphen zone Z
[]	Set column	<	Set left tab <
D	Delete character	>	Set right tab >
G	Get saved ruler	.	Set decimal tab .
L	Set left margin	-	Clear tab stop -
R	Set right margin	?	Ruler Help Screen ?
I	Insert Blank		
Shift/	Clear all ruler settings		
<ESCAPE>	Embed ruler and exit		

MARK TEXT COMMANDS

(Insert = Ctrl/U; Edit = %)

U	Underline (solid)	B	Boldface ON
W	(broken)	C	Boldface OFF
D	(double)	P	Superscript ON
T	Strike-through	S	Superscript OFF
X	Underline OFF	S	Subscript ON S
		P	Subscript OFF

DELETING TEXT

D	Delete character	Q	Delete word
K	Delete line/ruler	Y	Delete sentence

TEXT INSERTION COMMANDS

~N	Insert text from keyboard
~I	Insert text from data file
~R	Insert text from list
~DN	Insert current date
~TN	Insert current time
~X	Output ESCAPE or CONTROL character to printer
~F	Output FONT commands to printer

OVERVIEW OPTIONS

(J)	Printer driver
(L)	Paragraph numbers
(M)	Manual sheet feed
(N)	Nopage
(P)	Print Spooler
(Q)	Letter quality
(Z)	Line numbers

BACKSLASH (\) COMMANDS

All of the following commands can be inserted in either uppercase or lowercase, but they must be preceded by the “\” (Backslash). These commands are actually acted upon by JET in the Overview or Printing mode. You will see them on the screen if the View is on. You can print them on a printer if you print using the (V) option.

*** {text}**

Specifies a comment line. Entire line is ignored during output. Must be used at the end of FOOTINGS and HEADINGS. However, it can be used anywhere to add a comment that you might want to see when in the Edit mode.

\ANCHOR *n*

Anchors, or starts printing on the current page at line “*n*”.

\BC START

\BC END

This is the two-column command. There are two ways of dealing with columns. The first is to use the \BC START and \BC END commands. With these commands you should first shorten the ruler to less than 40 characters and begin inputting your text. When it is printed, JET will split the text and print it in two columns. See also the \BC 1 and \BC 2 command.

\BC 1

\BC 2

The second way of dealing with multiple columns allows you to determine what text will be in each column. You again shorten your ruler to less than 40 characters, insert the \BC 1 command and begin entering the text. When you want to start the second column, insert the \BC 2 command and continue entering the text.

To turn off the two columns, insert another \BC 1 and change the ruler back to larger than 40 characters. The two columns will not appear while inserting or editing the document, but they will appear in the Overview mode.

\BEGINPAGE {*pagenumber*}

\BEGIN {*pagenumber*}

\BP {*pagenumber*}

Begins new page by printing optional footing on existing page, ejecting a page, and printing the optional heading on the next page. If the *pagenumber* parameter is specified, the page number counter is set to the specified number prior to printing the optional heading on the new page.

Make a habit of ending your document or letter with this command, as it will move the paper in your printer to a new page, ready for the next printout. If you don't use the \BP, the next printout will begin at the end of the last printed piece.

\BIN *n*

Specifies which bin of a multi-bin sheet feeder to use to feed the next sheet of paper. The \BIN command should appear just before a \BP command.

An example would be \BIN 2, which specifies that the next sheet feed will be fed from bin number two. *Note:* This command is only effective when used with Q option (Letter Quality Printers).

\CENTER

\CENTER *n*

\CENTER START

\C

The CENTER command centers the next line of text between the left and right margins. Only appears in the overview and print modes. To center multiple lines of text, three options are available:

\CENTER <i>n</i>	Centers the following <i>n</i> number of lines
\CENTER START	Centers all text until the next \CENTER END command is encountered
\CENTER	When used just before a wrapped paragraph, each line of the paragraph is centered

\CHAIN {*filename*} *itemname*

\CH {*filename*} *itemname*

Transfers control to specified item and continues output processing. Control does not return to the source item that caused the transfer of control. If the *filename* parameter is omitted, the current file name is used.

\CHAPTER

The CHAPTER command causes a centered CHAPTER to appear on the page that contains the command. It also adds the number 1 to the first occurrence of the chapter command and increments the chapter number by 1 at each subsequent occurrence. The CHAPTER command also generates an entry in the Table of Contents and starts a new page.

\CRT

Directs subsequent text to the terminal, rather than the printer.

\ENDIF

Turns off the conditional printing activated with the \IFATT command.

\FOOTING {*L or R*}
{{*text*} {'options'} . . .}

\FOOTING OFF
\F

Outputs following line(s) at the bottom of each page. Note that the \FOOTING command must be terminated with a “comment line” (*) on the line immediately following the footing text.

The *L* or *R* parameter following the command indicates which pages will have page footings. For example, the command \FOOTING L will cause the footing to appear on the left hand pages only.

The OFF directive disables the previously defined FOOTING. Here are the options:

D Outputs system date
 I Outputs the current item-id
 P Inserts page number, right justified
 PN Outputs page number, left justified
 T Outputs system time and date

Here are some examples:

```
\FOOTING
This end into shredder first.    Page 'P'
\* end of footing
```

\GALLEY *n*

The GALLEY command is used with the \BC double column to specify the number of blank spaces to add between the columns. Spaces between the columns may be removed by specifying a negative number such as: \GALLEY -2. This removes 2 spaces from between the columns.

\HEADING {*L or R*}
{{*text*} {'options'} . . .}

\HEADING OFF
\H

Outputs following line(s) at the top of each page. Note that the \HEADING command must be terminated with a “comment line” (*) on the line immediately following the heading text.

The *L* or *R* parameter following the command indicates which pages will have page headings. For example, the command \HEADING R will cause the heading to appear on the left hand pages only.

The OFF directive disables the previously defined HEADING. Here are the options:

- D Outputs system date
- I Outputs the current item-id
- P Inserts page number, right justified
- PN Outputs page number, left justified
- T Outputs system time and date

Here are some examples:

```
\HEADING
    Profit & Loss Report 'TL'
\* end of heading
```

\HILITE {*character*} {*side*}

\HILITE OFF

\HL

Specifies that following lines are to be printed with the specified character two spaces into the margin(s). The default character is an asterisk (*). The OFF directive disables the HILITE function. Here are possible “Side” Choices:

- B Both left and right margins
- I Inside margins
- L Left margin
- O Outside margin
- R Right margin

\HILITE OFF

\HL OFF

Terminates \HILITE function.

\IFATT *n* = “*value*”

\IFATT *n* # “*value*”

Specifies a value to be met before the next paragraph or block will print. The *n* parameter represents the attribute number from the current item

being processed and the value may be any literal value enclosed in single or double quotes. In other words, if the data in the data attribute is equal to or not equal to the *value* in quotes, then the next paragraph will print. Otherwise, printing will resume after the next \ENDIF command. For example:

```
\IFATT 1 = 'BOZO'  
Dear Clown,  
\ENDIF
```

If attribute 1 consists of the word “BOZO”, then the salutation on the following line is printed. If the name is not BOZO, then the text after the \ENDIF command is output. *Note:* Nesting of \IFATT commands is not supported.

\IFENTER = *string* \CHAIN {*filename*} *item-id*

Used in help screens only to allow chaining together screens.

\INDENT *n*

Indents the left margin by the number of spaces indicated by *n* which can be negative.

\INDEX *indextext*{*indextext* . . .}

Places the string of characters specified in *indextext* in a file for later printing.

CAUTION: Before you can generate an INDEX, you must create a file to hold the indexed items. To generate an index you must use this command along with the text names you want to index on each page of your document. See also the \INDEX.FILE command. For example:

```
\INDEX lions,tiger-like,beasts of burden
```

This creates index entries for lions, tiger-like, and beasts of burden, along with the page number of where they occurred. Each item in the index may contain more than one page number if the INDEX command for that line was used on more than one page.

\INDEX.FILE *filename*

Specifies file name to use for accumulating index entries. This command is placed in your document automatically when you generate an index through the Front-End menu system.

\JUSTIFY

\J

Enables right-justification and line fill mode. This command causes the following text to be justified or lined up with the Right Margin (sometimes called an “even right margin”). This is the default for a document. Justification will only occur on those lines where automatic word wrap or hyphenation was used.

\LPI *n*

Specifies the line spacing, or number of lines per inch for letter-quality printers. The default is 6 lines per inch. This can only be used with the (Q) (letter quality) option in printing.

\LPTR

This causes the following text to be printed on the printer. See also the (P) print option and the \CRT command.

\NOJUSTIFY

\NJ

Disables right-justification mode. The default is Justify On. See also the \JUSTIFY command.

\PAGE {*pagenumber*}

\PN {*pagenumber*}

\NUMBER {*pagenumber*}

Resets *pagenumber* to specified number.

\PAGE LENGTH *n*

\LEN *n*

Overrides the page length for the output device as indicated with the most recently executed TERM command. Used only in printing and overiewing.

\PAUSE

Stops output until any key is pressed, then continues output. Used in conjunction with the \PROMPT command to temporarily halt spooling to the printer until a response is provided to the \PROMPT command and any key is pressed at the CRT. See also the (M) print option.

\PCLOSE

\PC

This command forces the SPOOLER to print the print file.

\PFILE *n*

\PF *n*

Routes subsequent print output to a different SPOOLER hold file. The initial default setting is 1. NOTE: These numbers are NOT spooler hold file entry numbers!

\PITCH *n*

\PI *n*

Specifies the number of characters per inch for proportional spacing. This function is available only when using the (Q) option on letter quality printers.

\PROMPT text

Outputs specified text to terminal screen as a prompt message for the next ~INPUT or \PAUSE command.

\READ {*filename*} *item-id*

\R {*filename*} *item-id*

Transfers control to the specified item-id and returns upon completion. If the *filename* parameter is omitted, the current file name is used.

\READNEXT

Retrieves the next item-id from the previously established list. Used in conjunction with the \SELECT and ~INSERT commands. Only here for upward compatibility with some Ultiword (Ultimate's version of JET) documents.

\SB

Inserted automatically during pagination pass to indicate a soft page break. It is inadvisable to insert these manually into a document.

\SECTION *n*

The SECTION command sets the section levels, and generates an entry in the Table of Contents. The *n* corresponds to the level to increment. For

example, when $n = 1$, the first section number is incremented, when n is 2, the second section number is incremented.

\SELECT *filename* “*item-id*” {“*item-id*”. . .}

Allows selection of item-id's for insertion into a document. Can be used in conjunction with the ~INSERT command, or with the \IFATT command. No selection criteria are supported. For example:

```
\SELECT CUSTOMER-FILE "1001" "1002"
```

CUSTOMER-FILE is the filename. “1001” and “1002” are the item-ids of the items to select.

\SETSECTION n

This command sets n as the number that appears at the beginning of the SECTION numbers. For example: if n were set to 6, the section becomes section 6. The \SETSECTION command must be immediately followed by a \SECTION command in a document.

\SPACING n

\SP n

Specifies the number of lines to skip between print lines. May be any whole or decimal number between 1 and 4. Spacing 1 is the default for single spacing.

Note that spacing may be set for half spacing between lines. Specify n as 1.5 for one and 1/2 spacing between the lines, or .5 for half spacing.

\TEST PAGE n

\TEST n

\TP n

Counts the number of lines remaining to be output on the current page. If the n parameter is less than the count returned, then the following text is output on the current page, otherwise, a page is ejected and the text is printed on the new page.

\TITLE

title text

Places following line of title text into the Table of Contents. *Caution:* Before you can generate a Table of Contents, you must create a file to hold the titled items. To generate a Table of Contents you must use this command

on a single line. The next line of text will be the entry that will be inserted into the Table of Contents.


\TITLE.FILE *filename*

Indicates the name of the file used to accumulate the Table of Contents. This command is placed in your document automatically when you generate a Table of Contents through the Front-End menu system.

\WINDOW *x1,y1;x2,y2*

Used inside help screens to draw a window, frame or box on a screen. It is inadvisable to include this command in documents. *x1* and *y1* are column and row coordinates of the top left corner. *x2* and *y2* are column and row coordinates of the bottom right corner.

TCL



This guide is intended to cover the “generic” version of Pick, including the Pick-on-the-AT (and XT) versions. Most of the commands listed in this section will work on most of the Pick systems in production today. Some commands, however, may behave differently than described in this text.

At the time of this writing there was no document from SMA on standard TCL commands. As a result, the only commands that contain a compatibility index are those that are unique to a specific release of the Pick system for the XT and/or AT. For example, many new verbs were added to release 2.2 for the AT and are not available on earlier releases or for the XT version.

STANDARD TCL OPTIONS

Options are special characters that alter the normal effect of a TCL command. They nearly always appear at the end of a TCL command. Options *must* be preceded by a left parenthesis character. The right parenthesis is optional, as are commas (,) between the parameters. This rule does not, however, apply to spooler options.

The following two options are available for use with most TCL commands:

- N Activates the NOPAGE function on output to the terminal
- P Directs output to the system printer, via the spooler

➡ About the Terms Used in This Section

Please refer to the glossary located at the back of the *Pocket Guide* for an explanation of the standard terms used in the PICK system, along with the special terms that were created during the production of this book.

SPECIAL CURSOR CONTROL FUNCTIONS

Control-(shift)-__<cr>

Line continuation character. Extends TCL command line by another 140 characters.

Control-H

Backspaces one character.

Control-I

Same as the TAB key.

Control-J

Same as a linefeed.

Control-Q

Issues an X-ON character, resuming the process previously suspended with an X-OFF (Control-S).

Control-R

Retypes last line.

Control-S

Issues an X-OFF character, which may be reversed with an X-ON (Control-Q).

Control-W

Backs up one word.

Control-X

Cancels current input line, or terminates a multi-page display, returning control directly to TCL.

TCL VERB CLASSES

TCL commands may be divided into three different categories according to how they are invoked:

➤ TCL-I Verbs

These verbs require only the verb and a <cr>. Most, however, allow options. The common denominator to TCL-I verbs is that they do not affect files. However, the CREATE-FILE, CLEAR-FILE, and DELETE-FILE verbs are considered TCL-I, even though they break the apparent rule of not affecting files. Here are some examples of TCL-I Verbs:

```
>WHO  
>TIME  
>POVF
```

➤ TCL-II Verbs

TCL-II verbs require a file name, and usually a list of one or more item-ids. The only time that they do not require a list of item-ids is when the verb follows a SELECT, SSELECT, QSELECT, or GET-LIST. To determine if a verb falls into this category, issue the LISTVERBS command. Any verb with a “2” in the second column (attribute) is considered a TCL-II verb and must conform to this syntax. Here are some examples of TCL-II Verbs:

```
>ED MD LIST
```

In this example, one item is requested from the file called MD.

```
>ED MD LIST SORT COUNT
```

Here, multiple item-ids are requested. Note that each item-id is separated by a space.

```
>ED MD *
```

The * is a special symbol used to request *all* items in a file.


```
>SELECT MD = 'LIST'
```

```
n ITEMS SELECTED.  
>ED.MD<cr>
```

In this final form, a select list is built with the SELECT verb. Note that the item-id list does not have to be specified.

➡ ACCESS Verbs

The third, and final class of verbs are ACCESS verbs. These are, by far, the most sophisticated of all verbs and require at minimum a verb and a file name. See the ACCESS section for further information. To determine if a verb belongs in this category, issue the LISTVERBS command. If the second column (attribute) contains a “35”, it’s an ACCESS verb. Here are some examples of ACCESS commands:

```
>SORT CUSTOMERS BY NAME NAME ADDRESS CSZ LPTR  
>SSELECT INVOICES WITH NO PAYMENT.AMOUNT BY DATE  
>SSELECT ORDERS BY SHIP.DATE
```

TCL COMMANDS

>:FILES

Initiates a full restore as an alternative to doing it from the “options” prompt. The proper media type must be indicated with the SET- FLOPPY or SET-SCT prior to using this command.

Compatibility: Pick 2.2 and higher

>:STARTSPOOLER

Please refer to the SPOOLER section for information on this command.

>:SWD *filename itemlist**

Switches E pointer(s) to D pointer(s).

>:SWE *filename itemlist**

Switches D pointer(s) to E pointer(s).

>:SWX *filename itemlist**

Switches D pointer to DX pointer. Extreme care should be exercised in using this command.

>:SWZ *filename itemlist**

Switches D pointer to DZ pointer.

>:TASKINIT *n{,m}*

Used to pre-allocate extra workspaces for use by the EXECUTE statement in PICK/BASIC. The *n* parameter indicates the number of process workspaces to pre-assign per port. The optional *m* parameter indicates the maximum nesting depth. This command can be useful to recover disk in the event of running out of disk space by using “0” (zero) as the number of workspaces; any workspaces already assigned will be returned to the overflow table. Each workspace takes up about 400 frames.

Note: this verb may not exist in all versions of Pick, but the code for it does. The usual verb definition item contains a *P* in attribute 1, and “13C” in attribute 2.

>ACCOUNT-RESTORE *acctname {(options)}*

Restores an individual account from either an ACCOUNT-SAVE tape or a FILE-SAVE tape. Tape must be positioned to the beginning of the (data) files section, or at the beginning of the target account. Process will prompt for name under which account was saved on the tape. See also the RESTORE-ACCOUNTS, SEL-RESTORE, ACCOUNT-SAVE and FILE-SAVE commands. Here are the options:

- A Indicates that tape is already positioned at correct account
- S Ignores tape label information

>ACCOUNT-SAVE

Activates PROC to save an individual account to magnetic tape. As of release 2.2, Pick no longer writes a file mark at the beginning of the tape. See chapter 15, “Changes to Streaming Cartridge Tape Handling”.

>ADDD *decnum decnum*

Adds two decimal numbers and displays result in decimal.

>ADDENDA

Activates PROC to display information about Operating System enhancements made by Pick Systems.

Compatibility: Pick AT/XT

>ADDENDUM *number*

Activates PROC to display information about a specific feature related to Operating System enhancements made by Pick Systems. Use this after the ADDENDA command.

Compatibility: Pick AT/XT

>ADDX *hexnum hexnum*

Adds two hexadecimal numbers and displays result in hexadecimal.

>B/ADD

>B/DEL

>B/UNLOCK

Three orphan verbs that still linger around the Pick System. For a historical note, they were among the very first verbs added to the system. Fortunately, their use has almost disappeared entirely except for the SET-FILE and DELETE PROCs, both of which reside in the PROCLIB file.

>BASIC *filename itemlist {(options)}**

Activates BASIC compiler for translation of source code into object code. Here are the options:

- A Displays object code generated by PICK/BASIC compiler
- C Compiles object code without end of line characters. This makes the PICK/BASIC debugger completely useless for debugging the program. Some implementations have changed this to mean "backward Compatible"
- E Outputs error lines only
- I Shows line numbers for code placed in the program by an INSERT or INCLUDE statement, when used in conjunction with the L option
- L Lists program as it compiles
- M Generates program map
- N Activates NOPAGE function on output to the terminal
- P Directs output to system printer, via spooler
- Q Activates PAGE mode on errors

- S Suppresses symbol table generation
- X Cross-references all variables and places entries in the BSYM file
- Y Provides compilation statistics, including: the start clock time in milliseconds, the size of the code in bytes, the size of the descriptor table, the size of the symbol table in bytes, the number of symbol table entries, and the end clock time in milliseconds

>BATCH {integer-number}

Displays and/or changes the batch setting parameter for the current process. Advice: leave this alone.

>BEEP

Activates PICK/BASIC program to continuously beep on a CRT until someone reaches over and slams down their fist on any of the keys.

Compatibility: Pick AT/XT

>BLOCK-PRINT *text* {(P)}

Outputs block letters of text string on terminal screen, or directs output to system printer, via the spooler, when used with the P option. The characters in the text string are defined in the BLOCK-CONVERT file. For example:

```
>BLOCK-PRINT "JOE'S" BAR (P)
```

>CAT {*filename*}

Activates PROC to display information about the compile date/time for a PICK/BASIC program file.

Compatibility: Pick AT/XT

>CATALOG *filename itemlist**

This creates a “verb” from a compiled BASIC program, by placing an entry in the current accounts’ Master Dictionary that allows execution of the program by entering the program name at TCL. It is required for any program defined as an external subroutine.¹ To activate a cataloged BASIC program:

```
>programname {(options)}
```

¹ Not true on every Pick implementation. Some versions do not require the program to be cataloged if it exists in the same file as the calling program.

It is not necessary to re-CATALOG a program each time it is compiled.

>CHARGE-TO *acctname*

Terminates *charges* accumulation on current process. Subsequent activity usage statistics are subsequently accumulated to the specified sub-account.

>CHARGES

Displays total time logged on and CPU activity statistics.

>CHECK-SUM

Please refer to the ACCESS section for information on this command.

>CHOO-CHOO

Entertainment for the bored.

>CLEAR-BASIC-LOCKS

Resets all the execution locks established by LOCK statements from PICK/BASIC.

>CLEAR-FILE (DATA *filename*{,*filename*})

>CLEAR-FILE (DICT *filename*)

Clears specified file of all items and retains primary file space area.

>CLOCK

Activates program to demonstrate operation of real-time clock.

Compatibility: Pick AT/XT

>COLDSTART

Activates the “COLDSTART” PROC (procedure). This is usually the last step in the system coldstart procedure.

>COLOR {*foreground*}{,*background*}/{*options*}

Sets foreground and/or background colors on the color graphics monitor. The options allow special visual attributes to be activated or deactivated. May only be used from port 0 (zero). The following are valid color choices:

BLACK	GREEN
BLUE	MAGENTA
BROWN	RED
CYAN	WHITE

The following are valid options for visual attributes:

/B or /BLINK	Activate blinking
/F or /FULL	Full intensity foreground
/H or /HALF	Half intensity foreground
/NB or /NOBLINK	Deactivate blinking
/NR	Deactivate reverse video
/NOREVERSE	Deactivate reverse video
/R or /REVERSE	Activate reverse video

Compatibility: Pick AT/XT

>COMPARE *filename itemlist** {(options)}
WITH:{(*filename*) *itemlist*}

Produces comparative listing of specified *itemlist*, indicating where differences occur between the source and the target item(s). Here is a list of options:

- A All elements on each attribute. Default is first three elements, delimited with blanks (assembler source general format)
- N Activates NOPAGE function on output to the terminal
- P Directs output to system printer, via spooler
- S Suppresses display of identical items
- T Uses tape as source file. Must be in T-DUMP format and positioned at the correct file
- Z Outputs errors only, in assembly format

Status Characters:

- D Indicates deleted line
- I Indicates inserted line

>COMPILE² *filename itemlist** {(options)}

Activates BASIC compiler for translation of source code into object code. Synonym for BASIC verb. Here is a list of options:

² May not exist on all Pick Systems; use the BASIC verb instead.

- A Outputs assembled BASIC object code
- C Compiles object code without end of line characters
- E Outputs error lines only
- I Shows line numbers for code placed in the program by an INSERT or INCLUDE statement, when used in conjunction with the L option
- L Lists program as it compiles
- M Generates program map
- N Activates NOPAGE function on output to the terminal
- P Directs output to system printer, via spooler
- Q Activates PAGE mode on errors
- S Suppresses symbol table generation
- X Cross-references all variables and places entries in the BSYM file
- Y Provides compilation statistics, including: the start clock time in milliseconds, the size of the code in bytes, the size of the descriptor table, the size of the symbol table in bytes, the number of symbol table entries, and the end clock time in milliseconds

>COPY *filename itemlist {(options)}**
TO: {(filename) {itemlist}}

Copies specified *filename* and *itemlist* to a different file name and/or item list. Note: the file name may refer to a dictionary level by preceding the *filename* with the word, "DICT", followed by a space; no *itemlist** is required when the COPY follows a SELECT, SSELECT or QSELECT. Here is a list of options:

- n Integer number. Specifies the number of items to copy. Typically used for copying data for test files
- A Activates assembly MLIST format
- D Deletes source item after copy
- F Outputs a form-feed between each item
- I Suppresses item-id list during a file copy
- L Changes item to list format, if destination file is a "DC"-type file
- M Activates Macro (assembly) format on terminal or printer output only
- N Activates NOPAGE function on output to the terminal
- N Inhibits new items on copies between files. This means that the item-id must already exist in order for an item to be copied
- O Overwrites existing items when duplicate item-ids exist
- P Directs output to system printer, via spooler
- S Suppresses display of line numbers on copy to terminal or printer
- S Suppresses error message output on a file copy
- T Directs output to the terminal
- X Outputs in hexadecimal

>COPY-LIST *itemlist {(options)}**
TO:({filename} {listname})

Copies a previously saved *listname* to either a new name, a new file name, or the specified output device. Defaults to POINTER-FILE for file name. Here is a list of options:

- D Deletes item from source file after copy
- N Activates NOPAGE function on output to the terminal
- O Overwrites duplicate itemname(s)
- P Copies list to system printer, via spooler
- T Copies list to terminal
- X Outputs list in hexadecimal format

>COPYDOS DOSpath {(options)}

Copies data, in Pick storage format, from a DOS partition into the Pick partition.

Addressing the DOSpath:

drive:\filename

or

drive:\directory\filename

or

drive:\directory\directory\filename

The pattern continues. Effectively, there could be many directories. Here is a list of options:

- D Displays diagnostic information during translation
- F Flag characters. System prompts for further information
- L Creates list-type (indirect) item (Release 2.2)
- M Create multiple (Pick) items
- O Overwrite (duplicate) Pick item(s)
- P Strips most significant bit off characters as they are converted, except for flagged and/or translated characters (Release 2.2)
- R Indicates DOS data is in a random file
- S Indicates DOS data is in a sequential file
- T Translate characters. System prompts for further information
- X Creates hexadecimal image of DOS file

When file type (S or R) is omitted, the default is S. Note that when this is executed without the T and F options, DOS end-of-line characters (carriage returns or X'0D') are automatically converted to attribute marks (X'FE'). Line feeds (X'0A' and nulls X'00') are deleted.

Note: A verb called COPYPICK is now provided by Pick Systems. This verb allows the movement of Pick items into a DOS partition and must be executed from DOS. See Section 15, under the sub-heading "DOS Considerations" for more information on how to use this command.

Compatibility: Pick AT/XT

>COUNT

Please refer to the ACCESS section for information on this command.

>CREATE-ACCOUNT

Activates PROC to create new account and place entry in SYSTEM file. May be executed only from the SYSPROG account. After execution of the CREATE-ACCOUNT command, the following prompts appear:

ACCOUNT NAME?

The name of the new account. It must *not* already exist in the SYSTEM file. If you are unsure whether or not it already exists, end this procedure and issue the command:

>SORT ONLY SYSTEM<cr>.
L/RET CODES?

Retrieval code(s) for read-access to files. Typically left null.

L/UPD CODES?

Update code(s) for write-protection to files. Typically left null.

PASSWORD?

Optional logon password. See also the (TCL) PASSWORD command to change or remove passwords.

PRIVILEGES?

Privilege level. Must be either SYS0 (lowest), SYS1, or SYS2 (highest). As of release 2.2, SYS0 is now the default.

MOD, SEP?

Optional modulo and separation for new account MD. Typically, if omitted, this defaults to 29 for modulo and 1 for separation.

Upon answering this last prompt, the process copies all of the items from the NEWAC file into the new MD. The account is now ready for use. See the LOGTO command.

```
>CREATE-FILE filename dictmod{,dictsep} . .  
. . datamod{,datasep}  
>CREATE-FILE DICT filename mod{,sep}  
>CREATE-FILE DATA filename{,filename} mod{,sep}
```

Reserves space for and creates new *filename* placing file pointer entry in current accounts Master Dictionary (or dictionary when creating a DATA section on an existing dictionary). The modulo should be a prime number and may be up to 32,267 provided that the separation is one (1). The following are variable definitions:

mod Modulo. The number of groups to allocate for primary file space.
sep Separation. The number of frames per group. The default separation is one.

The following are examples Of The CREATE-FILE Command:

```
>CREATE-FILE CUSTOMERS 7,1 101,1
```

Creates both a new dictionary and data section.

```
>CREATE-FILE DICT POINTER-FILE 31,1
```

Creates a dictionary-only file.

```
>CREATE-FILE DATA CUSTOMERS,ARCHIVE 101,1
```

Creates a new data section called “ARCHIVE” and places a pointer to it in the dictionary of CUSTOMERS. The new data section may now “share” all the attribute-defining-items in the CUSTOMERS dictionary.

```
>CROSS-INDEX3 filename itemlist* {(options)}
```

Produces cross-reference of assembly source code in specified *filename*. Requires CSYM file. Here is a list of the options:

³ May not exist on all systems. Usually provided with the Assembly account.

- F Prompts for filename. Default is CSYM.
- O Scan operand field.

>CS

Clears the terminal screen.

Compatibility: Pick AT/XT

>CT *filename itemlist** {(options)}

Copies specified *itemlist* to terminal. See the COPY command for available options.

>DCD {*portnumber*}

Displays the current status of the Data Carrier Detect (DCD) on a specified port, or for the current port if one is not specifically requested. DCD is used to sense changes in the signal being transmitted over the wire to the terminal; when it drops, the terminal is automatically logged off. Note that special cabling is required prior to enabling this feature on a line. (Release 2.2)

Compatibility: Pick 2.2 and higher

>DCD-OFF {*portnumber*}

Disables the monitoring of Data Carrier Detect on the specified *portnumber*. If the *portnumber* is omitted, the current port is affected. May not be issued on or against line 0 (zero). The default setting is off.

Compatibility: Pick 2.2 and higher

>DCD-QN {*portnumber*}

Enables the monitoring of Data Carrier Detect on the specified *portnumber*. If the *portnumber* is omitted, the current port is affected. May not be issued on or against line 0 (zero). The default setting is off.

Compatibility: Pick 2.2 and higher

>DECATALOG *filename itemlist**

Deletes object code and catalog pointer if one exists in the MD of the current account.

>DEFINE-TERMINAL

Allows the addition or modification of term types for using different types of terminals on the system. All information needed for the process is requested from the menu presented upon issuing the command. See also the TEST-CURSOR command and section 15 for additional terminal characteristic features added to this command in release 2.2.

Compatibility: Pick AT/XT

>DEL-ACC

PICK/BASIC program used by the DELETE-ACCOUNT PROC.

>DELETE *filename* {*itemlist}**

Deletes specified *itemlist* from specified file.

>DELETE-ACCOUNT<cr>

Account Name?account name

Deletes all files in account and removes account name from SYSTEM file. Should be executed from the SYSPROG account with all other users logged OFF. After all of the files to be deleted have been listed, the user is asked if the account should still be deleted. This requires a Yes or No response prior to deletion.

>DELETE-FILE *filename*

>DELETE-FILE DICT *filename*

>DELETE-FILE DATA *filename*{*filename*}

Releases all frames used by specified file to overflow table, and removes *filename* from current account's Master Dictionary. The DICT (dictionary) level may not be deleted without first deleting any data sections that it may have pointers to.

>DELETE-LIST *itemlist**

Removes the specified *listname*{s} from the POINTER-FILE.

>DIVD *decnum decnum*

Divides two decimal number and displays result in decimal.

>DIVX *hexnum hexnum*

Divides two hexadecimal numbers and displays result in hexadecimal.

>DTR (*radix*) *value*

Converts decimal value to specified radix (base) and displays result. If the radix is omitted, a radix of 16 is assumed, and this behaves like the DTX command, converting decimal to hexadecimal. For example:

```
>DTR 2 8  
1000
```

This converts the decimal number 8 to its binary (base 2) equivalent.

>DTX *decnum*

Converts decimal value to its equivalent hexadecimal value.

>DUMP {*.*}*fid*{*-fid*} {*Spooler Options Handler*}

Outputs contents of specified fid(s). A fid specification preceded by a period (.) indicates that the address format is in hexadecimal. Here is a list of options:

- n Specifies individual frame.id
- n-n Specifies beginning and ending frame.id range
- A Displays output in EBCDIC format
- C Dumps entire frame, beginning with byte zero
- G Outputs each sequentially linked frame in group
- L Outputs forward frame linkages
- N Activates NOPAGE function on output to the terminal
- P Directs output to system printer, via spooler
- U Outputs backward frame linkages
- X Outputs in hexadecimal and character format

>ECHO

Toggles terminal echo. When terminal echo is off, characters entered from the keyboard are not displayed on the terminal screen.

>ED *filename itemlist {(*options*)}**

>EDIT *filename itemlist {(*options*)}**

Activates editor processor for entry or update of programs, PROCs and data items. Here is a list of options:

- A Activates Assembly formatter. Equivalent to AS command
- M Activates Macro expansion function. Equivalent to M command
- P Directs output to system printer, via spooler
- S Suppresses line numbers, in normal edit mode, or suppresses object code display when the assembly formatter is “on”. Equivalent to S command
- Z Suppresses “Top” and “EOI” messages

>EDIT-LIST *listname* {(Z)}

Retrieves specified *listname* and enters editor processor. Refer to the EDITOR section for the available commands and syntax requirements. Here is the option:

- Z Suppresses “TOP” and “EOI” messages

>EPSON

Activates PICK/BASIC program to change characteristics for an EPSON printer.

Compatibility: Pick AT/XT

>EXCHANGE *filename itemname1 itemname2*

Activates PROC to switch the names of the two specified items.

>FC

Displays the current status of “flow control” or Data Set Ready (DSR) and X-ON/X-OFF protocol on a specified port, or for the current port if one is not specifically requested. When the DSR signal is off or the X-OFF protocol is invoked, the output to the port is suspended.

Compatibility: Pick 2.2 and higher

>FC-OFF {*portnumber*}

Disables the flow control on the specified *portnumber*. If the *portnumber* is omitted, the current port is affected. May not be issued on or against line 0 (zero). The default setting is on. See also the FC-ON, MODEM-ON and MODEM-OFF commands.

Compatibility: Pick 2.2 and higher

>FC-ON {*portnumber*}

Enables the flow control on the specified portnumber. If the portnumber is omitted, the current port is affected. May not be issued on or against line 0 (zero). The default setting is on. See also the FC-OFF, MODEM-ON and MODEM-OFF commands. (Release 2.2)

Compatibility: Pick 2.2 and higher

>FDISK

Activates menu to allow any of the following functions:

- ☐ Change partition pointer. This affects the boot process next time it is used.
- ☐ Delete the Pick partition.
- ☐ Display partition information.
- ☐ Create a PICK Partition.
- ☐ Select next fixed disk drive. (Release 2.2)
- ☐ Undo changes to partition data, leaving them the way they were prior to invoking FDISK. (Release 2.2)

This command may only be issued from the SYSPROG account while on port 0 (zero). See also the REBOOT command.

>FILE-SAVE

Activates PROC to save the entire contents of the system on magnetic tape, typically followed by a T-DUMP of the STAT-FILE and generation of the file statistics report. See also the ACCOUNT-SAVE, LIST-FILE-STATS, and SAVE commands.

As of release 2.2, Pick no longer writes a file mark at the beginning of the tape. See section 15, "Changes to Streaming Cartridge Tape Handling".

>FILECOMP

Activates PICK/BASIC program to compare and analyze the contents of two files. Pick Systems (the company) advises reviewing the program itself prior to using it, as it requires having a "special" file established.

Compatibility: Pick 2.2 and higher

>FIND

Activates PICK/BASIC program to search a file for the existence of one or more strings of characters in any attribute and to optionally save the items in a saved list.

Compatibility: Pick AT/XT

>FKEYS

Activates PICK/BASIC program used for maintaining function key definition items. Pick Systems advises reviewing the program in the SYSPROG-BP file prior to using it.

Compatibility: Pick 2.2 and higher

>FORMAT

Activates process to format floppy diskettes for use ONLY with the Pick system. This command may only be issued from the SYSPROG account. Note that as of release 2.2, the FORMAT command now supports 5-1/4 and 3-1/2 inch diskettes.

Compatibility: Pick AT/XT

>GET-LIST {*filename*} *listname*

Retrieves *listname* from specified file, or defaults to POINTER-FILE if no filename is specified.

>GROUP *filename* {(*options*)}

Outputs hashing information on specified *filename*. Output includes base *fid*, the item-id(s) in each group, each items' (hexadecimal) byte count field, a count of the total number of items in each group, the total number of bytes, the total number of "full" frames, and the number of bytes used in the "last" frame of each group. Here is a list of options:

- I Suppresses display of null groups
- N Activates NOPAGE function on output to the terminal
- P Directs output to system printer, via spooler
- S Suppresses output of item-id's

>HASH-TEST

Please refer to the ACCESS section for information on this command.

>HUSH

Toggles terminal output on or off. When this is activated, the output from any process will not appear on the screen. Same as the HUSH command.

>INTER *{integer-number}*

Displays and/or changes the interactive setting parameter for the current process. Advice: leave this alone.

>ISTAT

Please refer to the ACCESS section for information on this command.

>ITEM *filename itemlist* {options}*

Outputs the base fid of the group to which the specified item-id “hashes”, and a list of all item-id’s that are currently “hashed” to the same group. Also displayed is each items’ (hexadecimal) byte count field, a count of the total number of items in the group, the total number of bytes, the total number of “full” frames, and the number of bytes used in the “last” frame of the group. Here is a list of options:

- N Activates NOPAGE function on output to the terminal
- P Directs output to system printer, via spooler
- S Suppresses output of item-id’s

>LINK-WS

Relinks the process workspace for all lines. This occurs automatically during a coldstart and normally is not required.

>LIST

Please refer to the ACCESS section for information on this command.

>LIST-FILE-STATS

Activates PROC to print the file statistics report from the data placed in the STAT-FILE by the most recently executed FILE-SAVE or ACCOUNT-SAVE. See also the SAVE command.

>LIST-ITEM

Please refer to the ACCESS section for information on this command.

>LIST-LABEL

Please refer to the ACCESS section for information on this command.

>LIST-LOCKS {(options)}

Outputs base fid and line location of all current group locks, BASIC execution locks, and optionally, item locks (indicated by an asterisk “*”). Here is a list of options:

- n Displays item-lock status for the group starting at the specified frame-id (fid)
- I Displays item-lock status
- P Directs output to system printer, via spooler

>LIST-PORTS {(Z)}

Displays the baud rate, parity bits, stop bits and word length for every connected port on the system. See also the SET-PORT command. Here is the option:

- Z Displays the status for all configured ports, connected or not

Compatibility: Pick 2.2 and higher

>LISTABS

Please refer to the SPOOLER section for information on this command.

>LISTACC {acctname} {NOPAGE} {LPTR}

Outputs system usage statistics from ACC file for all accounts, or individually specified account.

>LISTCONN {filename} {NOPAGE} {LPTR}

Outputs connectives in specified *filename* (default file name is MD).

>LISTDICT {filename} {NOPAGE} {LPTR}

Outputs all dictionary item information in specified *filename* (default file name is MD).

>LISTFILES {filename} {LPTR}

Outputs all file names in specified *filename* (default file name is MD).

>LISTPEQS

Please refer to the SPOOLER section for information on this command.

>LISTPROCS {*filename*} {NOPAGE} {LPTR}

Outputs all PROCs in specified *filename* (default file name is MD).

>LISTPTR

Please refer to the SPOOLER section for information on this command.

>LISTU

>LISTUSERS

Outputs port number, account name, PCB-FID, logon time, logon date, and location of every terminal currently logged onto system.

>LISTVERBS {*filename*} {NOPAGE} {LPTR}

Outputs all verbs in specified *filename* (default file name is MD).

>LOCK-FRAME {*.*}*fid*

Locks specified *fid* in memory. See also UNLOCK-FRAME.

**>LOGOFF {(U)} <cr>
CHANNEL:*line#***

Logs off process on specified portnumber. The option is:

U Unconditional logoff. Prevents cleanup of work areas.

**>LOGON {(P)}<cr>
CHANNEL, NAME:*line#*,*acctname***

Logs on specified *line#* to specified account name.

As of release 2.2, a change has been made to the way the logon process is handled. After three unsuccessful logon attempts, the port is rendered “inactive”, preventing further attempts. To “re-activate” the port, press any key (other than the <cr>) followed by two <cr>’s. The option is:

P Starts process as a “phantom” line. Note that this does not have the same meaning as what is often considered a “phantom”, where the process spawns a new process workspace. Rather, this option requires an existing unused port, just like any other port being used for a remote logon. In this context, “phantom” means that it is treated like a spooler process. This means that the logon sequence is suppressed,

which turns off the display of the time and date logged on, the “welcome” message” and the automatic update of the ACC file.

>LOGTO *acctname*{*password*}

Terminates current session, starts new session on specified *acctname*. If *password* is omitted and required, the system will prompt for it.

>LOOP-ON *command*

Starts an infinite repetition of the specified TCL command.

>MESSAGE (or MSG) *acctname text*

>MESSAGE (or MSG) * *text*

>MESSAGE (or MSG) *!line# text*

>MESSAGE (or MSG) *!* text*

Sends message to all users logged on to the specified *acctname*, or, if the literal, asterisk (*), is used, to all users currently logged on to system.

The *!line#* form of the command sends a message to the specified line-number. For example:

>MESSAGE !0 ARE YOU THERE?<cr>

Sends the message to port zero only.

The *!** form sends the message to every terminal (and serial printer!) connected to the system, whether or not the device is logged on.

The “receiver(s)” of the message see the time/date, account name/port number of the transmitter, and the text of the message.

>MLIST *filename itemlist* {(options)}

Produces formatted assembly source code listing of specified *itemlist*. Here is a list of options:

- n** Specifies line number to list
- n-n** Specifies beginning and ending line numbers of listing
- E** Outputs error lines only
- M** Outputs macro expansions
- N** Activates NOPAGE function on output to the terminal
- P** Directs output to system printer, via spooler
- S** Suppresses object code display
- Z** Inhibits entry into editor when used with E option

>MLOAD *filename itemlist {(options)}**

Loads assembled object code into virtual memory address defined in routine. Here is a list of options:

- N Returns item checksum data without actually loading item
- V Verifies mismatches and errors only

>MODEM-OFF

Disables the flow control on the current port. May not be issued on or against line 0 (zero). The default setting is on. See also the FC, FC-ON, FC-OFF, and MODEM-ON commands.

Compatibility: Pick 2.2 and higher

>MODEM-ON

Enables the flow control on the current port. May not be issued on or against line 0 (zero). The default setting is on. See also the FC, FC-ON, FC-OFF, and MODEM-OFF commands.

Compatibility: Pick 2.2 and higher

>MONO {/options}

Sets options for the monochrome display unit. The options allow special visual attributes to be activated or deactivated. May only be used from port 0 (zero). Here are the valid options for the visual attributes:

/B or /BLINK	Activate blinking
/F or /FULL	Full intensity foreground
/H or /HALF	Half intensity foreground
/NB or /NOBLINK	Deactivate blinking
/NU	Deactivate underlining
/NOUNDERLINE	Deactivate underlining
/NR	Deactivate reverse video
/NOREVERSE	Deactivate reverse video
/R or /REVERSE	Activate reverse video
/U	Activate underlining
/UNDERLINE	Activate underlining

Compatibility: Pick AT/XT

>MSG

See the MESSAGE command.

>MULD *decnum decnum*

Multiplies two decimal numbers and displays result in decimal.

>MULX *hexnum hexnum*

Multiplies two hexadecimal numbers and displays result in hexadecimal.

>MVERIFY *filename itemlist *{{options}}***

Compares assembled object code in virtual memory against corresponding itemlist in specified filename. Here is a list of options:

- A Outputs columnar listing of all mismatches
- E Outputs errors only
- P Directs output to system printer, via spooler

>OFF

Terminates current session, logs current process off system.

>OKIDATA

Activates a menu to allow the changing of printer characteristics for the Okidata parallel printer.

Compatibility: Pick AT/XT

>P

Toggles terminal output on or off. When this is activated, the output from any process will not appear on the screen. Same as the HUSH command.

>PACK

Activates PICK/BASIC program to compact multiple items into composite items for transmission. Pick Systems provides this program, but no support for it. See also the UNPACK command.

Compatibility: Pick 2.2 and higher

>PASSWORD

Allows adding, changing or removing an existing account password. Must be issued from the SYSPROG account. The password is encrypted into an 8-byte hexadecimal string. It is not necessary to know what the old password is in order to change or remove it.

>POVF {(P)}

Outputs available linked and contiguous overflow space.

>POWER-OFF

Brings the system to an “orderly shutdown” by flushing memory to disk, disabling all users, and parking the disk drive heads prior to halting the system. May only be executed from port 0 (zero) when all other users are logged off. As of release 2.2, this command will not work if any lines other than zero are active. This includes spooler-controlled lines (printers).

Compatibility: Pick AT/XT

>PRINT-ERR *filename itemlist**

Outputs text of specified error message number.

>PRIO

Displays the batch, interactive and the timeslice parameters for the entire system.

>PVERIFY *filename itemlist**

Verifies integrity of PICK/BASIC object code. Hasn’t been needed since about 1980 (R80).

>QSELECT *filename itemlist {(attr#)}**

Creates a select list item from the specified *filename* and *itemname* entered, similar to the GET-LIST command. If the optional *attr#* (attribute number) is specified, the list is created from the multivalues contained in the specified attribute number.

>REBOOT

Causes the system to re-initialize, rather than using the CTRL-ALT-DEL keys. May only be issued from port 0 (zero). Disables all users and flushes memory prior to rebooting. See also the FDISK and POWER-OFF commands and section 15 for more documentation on booting Pick. As of release 2.2, this command will not work if any lines other than zero are active. This includes spooler-controlled lines (printers).

Compatibility: Pick AT/XT

>RECOVER-FD<cr>
ITEM NAME:*itemname*<cr>

Allows recovery of an item deleted via an FD command. It must be executed immediately after the FD command, otherwise the item is lost. If a list of items is active, there is no chance to recover the item.

>REFORMAT

Please refer to the ACCESS section for information on this command.

>RESTORE-ACCOUNTS

Activates PROC to restore multiple accounts from a FILE-SAVE tape or diskette(s). See also the ACCOUNT-RESTORE command.

Compatibility: Pick 2.2 and higher

>RTD {*radix*} *value*

Converts radix (base) to its decimal equivalent and displays result. If the radix is omitted, a radix of 16 is assumed and this behaves like the old XTD command, converting hexadecimal to decimal. For example:

```
>RTD 2 1000
8
```

This converts the (binary — base 2) number 1000 to its decimal equivalent.

>RUN *filename itemname* {(options)}

Executes a PICK/BASIC program. Here is a list of options:

- A Prohibits entry into PICK/BASIC debugger; aborts on error conditions
- D Enters PICK/BASIC debugger prior to execution
- E Enters debugger on any error condition
- I Inhibits variable initialization
- N Activates NOPAGE function, on output to the terminal
- P Directs output from PRINT statements to system printer, via spooler
- S Suppresses run-time warning messages (Useful when demonstrating new programs)

>RUNOFF *filename itemlist {(options)}**

Activates output function of RUNOFF text processor. Here is a list of the options:

- n Any integer. Specifies the number of times to overstrike a character or characters designated as boldface
- C Suppresses .CHAIN and .READ output
- I Outputs next item name
- J Suppresses highlighting function
- N Activates NOPAGE function on output to the terminal
- P Directs output to system printer, via spooler
- S Suppresses boldface and underline function
- U Outputs in all upper case

>S-DUMP

Please refer to the ACCESS section for information on this command.

>SAVE {(options)}

Activates the save (backup) processor for any of the following functions:

- 1 Copy contents of entire system to magnetic tape
- 2 Copy contents of a single account to magnetic tape
- 3 Produce file statistics data.

See also the ACCOUNT-SAVE, FILE-SAVE, and LIST-FILE-STATS commands. Here is a list of the options:

- D Saves data files
- F Outputs filenames. When omitted, only SYSTEM-level names are displayed
- G Handles GFE's encountered during SAVE. Rather than stopping and waiting for a response, it skips the rest of the current group and proceeds to next group. Logs entries in STAT-FILE when the S option is used
- I Individual account (used in ACCOUNT-SAVE)
- N No overflow space is required
- P Outputs filenames to system printer
- S Creates file statistics in STAT-FILE
- T Outputs to magnetic media. Omitting this option is useful for re-generating system statistics and/or finding GFE's. Hint: don't use the G option when trying this

>SAVE-LIST {*filename*} *listname*

Saves *listname* in specified *filename*. The default *filename* is POINTER-FILE.

>SEL-RESTORE *filename itemlist* (options)*

ACCOUNT-NAME *?:acctname*

FILENAME *?:filename*

Selectively restores items from either a FILE-SAVE tape or an ACCOUNT-SAVE tape. Tape must be positioned at the start of the file (data) section on a FILE-SAVE tape, or at the beginning of the account on an ACCOUNT-SAVE tape.

The specified *acctname* and *filename* must match exactly with the appropriate name on tape. The first file *filename* reference specifies the file in which the items are to be restored. The latter reference to *filename* indicates the file name, as stored on the magnetic tape. Entering a <cr> at the latter *filename* reference specifies to restore the specified account's MD (Master Dictionary).

Note that the term, *tape* is used for a documentation convention only. The type of media is determined prior to using this command by the use of either the SET-FLOPPY, SET-SCT or other media-attaching commands. After the media is attached, all of the normal tape-handling verbs may be used, with a few exceptions like T-RETEN and T-ERASE. Here is a list of the options:

- A Indicates that tape is already positioned at the correct account
- C Allows restoration of every item before next end-of-file, when used with the N option
- F Displays file names for all accounts. May not be used with the N option
- I Suppresses display of item-id's
- N Identifies file by location (order) number
- O Overwrites items when duplicate item-id's exist
- S Suppresses "EXISTS ON FILE" messages when the target item-ids already exist and the O option is not being used

>SELECT

Please refer to the ACCESS section for information on this command.

>SET-BAUD {*portnumber*,}*baud rate*

Allows changing the baud rate for lines 1 and 2 on the XT, and for 1 through 10 on the AT. The default baud rate for each line is 9600. When the line (*port*) number is not specified, the current line is affected. See also the SET-PORT command. The valid baud rate choices are:

110, 150, 300, 600, 1200, 2400, 4800, 9600
(and 19200 on release 2.2 and higher).

Compatibility: Pick AT/XT

>SET-DATE *mm/dd/yy*{*yy*} {(U)}

Sets the system date to specified setting. See also the SET-DATE-EUR and SET-DATE-STD commands. Note that this is now a PROC that executes the verb called SET.DATE. Here is the option:

U Updates the hardware real-time clock, as well as the system (software clock)

Compatibility: Pick AT/XT

>SET-DATE-EUR

Toggles the standard date format to European format : *dd/mm/yy*.

Compatibility: Pick AT/XT

>SET-DATE-STD

Toggles the European date format to North American format : *mm/dd/yy*.

Compatibility: Pick AT/XT

>SET-FILE {*acctname* {*filename*}}

Creates or updates the item called QFILE in the MD of the current account. Provides access to specified *filename* in specified *acctname* by using the file name, QFILE, in place of the *filename* parameter during any file-handling commands.

>SET-FLOPPY {(*density,drive*)}

Used after the T-ATT command to designate the density to read/write diskettes and which floppy drive to use, when more than one is present. Drive choices are typically A or B. See also the FORMAT command. Don't forget to "rewind" the diskette with the T-REW command prior to attempting any read or write operation. Usually, this doesn't take too long. The defaults are : drive A, High density.

As of release 2.2, 3-1/2 inch floppies are supported. The density choices are:

S Standard density (360Kb for 5-1/4 inch, 720Kb for 3-1/2 inch)
H High density (1.2Mb for 5-1/4 inch, 1.44Mb for 3-1/2 inch)

Compatibility: Pick AT/XT

>SET-FUNC *filename itemname*

Specifies name of item to use in assigning function keys for use only by port 0 (zero). Each function key may be assigned one character. See the file called FUNCKEYS on the SYSPROG account. The default upon booting is that the function keys are undefined. See the item called SET-KBRD.DOC in the DOC file in the SYSPROG account for more information on defining function keys.

Compatibility: Pick AT/XT

>SET-KBRD *filename itemname*

Specifies name of item to use in assigning the type of keyboard for use only by port 0 (zero). The items available in the KEYBOARDS file on the SYSPROG account include : BRITISH (ENGLISH), DVORAK, FRENCH, FRENCH 2, GERMAN, ITALIAN, SPANISH, SPANISH 2 and USA (the default). See the item called KBRD.DOC, in the DOC file on the SYSPROG account. It contains a detailed explanation for constructing keyboard definition items. As of release 2.2, Pick Systems now provides support for 101/102 keyboards.

Compatibility: Pick AT/XT

>SET-LPTR

>SET-LPTR *number*

>SET-LPTR {*number*{,*delay*},{*burst*},{*slice*}}

Used to display or change performance parameters on parallel printers. To obtain an optimum blend of printer and system performance, experiment with different settings. When no parameters are provided, the current settings are displayed. A printer number may be specified to display its current settings. Like the TERM command, existing parameters may be left intact by providing two successive commas where the parameter would normally appear.

The *number* parameter indicates the printer number in the range 0 to three.

The *delay* parameter indicates the number of times to attempt to send a character to the printer. The higher the value, the slower the system performance, but faster printer performance. The default is 300, and the value must be in the range 1 to 4095.

The *burst* parameter indicates the number of characters to transmit to the printer in one packet. Relates to size of printer (RAM) buffer. The default is 200, and must be in the range 1 to 255. Pick Systems suggests that 15 provides good performance.

The *slice* parameter indicates the number of timeslices that the printer waits prior to receiving its next time slice. The default is 1, and it must be in the range 1 to 15. Pick Systems advises using a number equal to or greater than the number of users on the system.

Compatibility: Pick AT/XT

>SET-PORT {*portnumber*{*baud*,*parity*,*stopbits*,*wordlength*}}

Activates program to display or change baud rate, parity bits, stop bits and word length for a port. When no *portnumber* is specified, the existing settings are displayed.

The valid baud rates (as of release 2.2) are: 110, 150, 300, 600, 1200, 2400, 4800, 9600, and 19200. The default baud rate is 9600.

To leave an existing parameter intact, use two successive commas (like the TERM command). Parity may be N for none, O for odd or E for even. *Stop bits* may be either 1 or 2. *Word length* may be either 7 or 8. See also the LIST-PORT command.

Compatibility: Pick AT/XT

>SET-SCT (*blocksize*)

Used after the T-ATT command to designate that the Streaming Cartridge Tape should be used for any subsequent “tape” operations. If the *blocksize* is omitted, it will default to 16384. The block size must be between 2048 and 16384, and a multiple of 512.

Compatibility: Pick XT only

>SET-SCT {*buffers*} {{{*blocksize*}{R}}}

Used after the T-ATT command to designate that the Streaming Cartridge Tape should be used for any subsequent “tape” operations.

The *buffers* parameter indicates the number of 512-byte buffers to assign for the tape buffer. The default (and minimum) is 128. The maximum is a function of how much memory is present on your system. Up to half of the memory may be allocated. For instance, if your system has 640Kb of main memory, the maximum number of buffers is 640.

If the *blocksize* is omitted, it defaults to 16384. The block size must be between 512 and 32500, and be a multiple of 512. Here is an explanation of the option:

R Indicates that the tape is in “old” format, meaning that it was created on a version prior to 2.2

Compatibility: Pick AT 2.2 and higher

>SET-SYM *filename* {(T)}

Sets symbol file pointer for use by system debugger. *Filename* is typically PSYM. The option is:

T Sets secondary file pointer to TSYM file

>SET-TERM *termwidth,termdepth,lineskip,lfdelay, ffdelay,backspace,printwidth,printdepth, termtype* {(R)}

Specifies default terminal and printer characteristics for entire system. Any parameter may be left intact by entering a null (two consecutive commas). The *termtype* parameter may appear anywhere in the command. The following is a list of term definitions:

<i>termwidth</i>	Number of print positions per line on terminal
<i>termdepth</i>	Number of lines on terminal
<i>lineskip</i>	Number of blank lines at bottom of screen
<i>lfdelay</i>	Number of delay characters output after linefeed
<i>ffdelay</i>	Number of delay characters output after a top-of-form is executed
<i>backspace</i>	Decimal number of ASCII character to echo to the terminal when the backspace key is pressed
<i>printwidth</i>	Number of print positions per line on printer output
<i>printdepth</i>	Number of lines per page on printer output
<i>termtype</i>	Type of terminal (for BASIC and PROC cursor control) Terminal Types Supported: (Consult your vendor for additional types — this list was compiled from the types provided with the Pick XT and AT):

A	Adds 580	L	Lear Seagler
B	Ampex 210	M	Ampex 80
C	C Itoh VT52	P	Pertec 701
D	Datamedia	R	Adds Regent
E	Esprit	T	Televideo 910
G	IBM 3161/3163	V	ADDS Viewpoint
I	IBM Monochrome	W	Wyse 50

The option for SET-TERM is:

R Redisplays TERM parameters after entry

A note about the *ffdelay* parameter: This parameter also determines when to eject a page (or clear the screen) between output pages. The following settings illustrate this concept:

ffdelay Function

- 0 Suppresses page eject between output pages on both the terminal and the printer
- 1 Suppresses page eject on the terminal only
- 2 Normal mode. Any value greater than or equal to two will clear the screen (and eject a page on the printer) between each output page

>SET-TIME *hh:mm(:ss)* {(U)}

Sets the system time to specified setting. Note that this is now a PROC that executes the verb called SET.TIME. The option:

- U Updates the hardware real-time clock, as well as the system (software clock)

Compatibility: Pick AT/XT

>SETUP.SIO

This program was provided with release 2.0 to alter the parity, stop bits and word length for a port. As of release 2.2, it has been renamed SET-PORT.

Compatibility: Pick AT/XT

>SLEEP *numberseconds*

>SLEEP *untiltime*

Suspends further processing until specified time is exhausted or reached. Here are some examples:

>SLEEP 600<cr>

Puts the process to sleep for ten minutes.

>SLEEP 23:59:59<cr>

Puts the process to sleep until one second before midnight. (It's as close as you can get.)

>SLICE *timeslice*

Used to set the *timeslice* for all ports on the system. Timeslice must be in the range 1 to 255. This verb may or may not be on your system and is usually better left alone.

>SNAKE

More entertainment for the hopelessly bored.

>SORT

Please refer to the ACCESS section for information on this command.

>SORT-ITEM

Please refer to the ACCESS section for information on this command.

>SORT-LABEL

Please refer to the ACCESS section for information on this command.

>SP-ASSIGN

Please refer to the SPOOLER section for information on this command.

>SP-CLOSE

Please refer to the SPOOLER section for information on this command.

>SP-EDIT

Please refer to the SPOOLER section for information on this command.

>SP-KILL

Please refer to the SPOOLER section for information on this command.

>SP-OPEN

Please refer to the SPOOLER section for information on this command.

>SP-STATUS

Please refer to the SPOOLER section for information on this command.

>SP-TAPEOUT

Please refer to the SPOOLER section for information on this command.

>SREFORMAT

Please refer to the ACCESS section for information on this command.

>SSELECT

Please refer to the ACCESS section for information on this command.

>STARTPTR

Please refer to the SPOOLER section for information on this command.

>STAT

Please refer to the ACCESS section for information on this command.

>STOPPTR

Please refer to the SPOOLER section for information on this command.

>STRIP-SOURCE *filename itemlist** {(O)} DESTINATION FILE: *filename*

Removes assembled object code from item list in source file name, moving only object code to specified destination file.

>SUBD *decnum decnum*

Subtracts two decimal numbers and displays result in decimal.

>SUBX *hexnum hexnum*

Subtracts two hexadecimal numbers and displays result in hexadecimal.

>SUM

Please refer to the ACCESS section for information on this command.

>T-ATT {*blocksize*} {(U)}

Attaches tape unit and assigns blocksize to tape I/O buffer. If no parameters are specified and the tape is already attached, this command outputs the current attachment status. The default blocksize varies between Pick systems. See also the SET-FLOPPY and SET- SCT commands. The option is:

U Unconditionally attaches the tape unit. Not a good way to win friends and influence people.

>T-BCK {*number tape records*}

Moves tape backwards over specified number of tape records. When the number of tape records is omitted, the tape moves back over the last EOF mark and requires a T-FWD command.

>T-CHK *{{options}}*

Checks a magnetic tape file for parity errors. When the “A” option is omitted, only the current tape file is checked. Here is a list of the options:

- A Checks to end of data
- L Displays label data if one is found after an end-of-file mark (release 2.2 and higher)
- P Directs output to system printer, via spooler

>T-DET {U}

Detaches tape unit from current process. The option is:

- U Unconditionally detaches tape unit. This requires a SYS2 privilege level.

>T-DUMP

Please refer to the ACCESS section for information on this command.

>T-EOD *{{options}}*

Moves tape forward to end of data. The options are:

- L Displays label data if one is found after an end-of-file mark (release 2.2 and higher). If the tape record size is different, it will change automatically
- P Directs output to system printer, via spooler

>T-ERASE

Performs a re-tension and erases a Streaming Cartridge Tape.

Compatibility: Pick AT/XT

>T-FWD *{number tape records}*

Moves tape forward to next end-of-file, or the specified number of tape records.

>T-LOAD

Please refer to the ACCESS section for information on this command.

>T-RDLBL *{{reelnumber}}*

Reads tape label number from optionally specified (hexadecimal) reel number.

>T-READ *{{Options}}*

Reads a block of data from the magnetic tape unit attached to current process. The options are:

- A Converts EBCDIC format to ASCII
- P Directs output to system printer, via spooler
- n Indicates to read block "n"
- n-n Specifies beginning and ending block from current position
- X Outputs in hexadecimal

>T-RETEN

Re-tensions Streaming Cartridge Tape by moving all the way to the end of the tape, then rewinding. Recommended prior to using tape for any purpose. Automatically performed during FILE-SAVE and ACCOUNT-SAVE.

Compatibility: Pick AT/XT

>T-REW

Rewinds magnetic tape to load point.

>T-SPACE *(number tape records)*

Moves tape forward specified number of tape records, displaying each tape label as it is encountered.

>T-STATUS

Indicates the type and drive number of the currently attached offline storage device.

Compatibility: Pick AT/XT

>T-WEOF

Writes an EOF (End Of File) mark on magnetic tape.

>T-WTLBL

Writes a tape label on the attached media.

>TA {*portnumber*}

Displays status of type-ahead buffer on specified port number, or current port if omitted. See also the TA-OFF and TA-ON commands.

Compatibility: Pick 2.2 and higher

>TA-OFF {*portnumber*}

Disables type-ahead buffer on specified port number, or current port if omitted. See also the TA and TA-ON commands.

Compatibility: Pick 2.2 and higher

>TA-ON {*portnumber*}

Enables type-ahead buffer on specified port number, or current port if omitted. See also the TA and TA-OFF commands.

Compatibility: Pick 2.2 and higher

>TABS

>TABS (I) {*tabstop*{,*tabstop* . . .}

>TABS I (S)

>TABS (O) {*tabstop*{,*tabstop* . . .}

>TABS O (S)

Displays tab stops previously set during the Editor process or assigns new tabstop positions for input (I) or output (O). Note that output *tabstop* positions are useful only on printing terminals that have a physical tab facility. Don't use output tabstops on normal CRT's.

Input tabstops may be disabled by issuing the command >TABS I<cr>. Similarly, output tabstops may be disabled by issuing the command >TABS O<cr>.

The S parameter indicates that the previous tabstop settings are to be used.

**>TERM *termwidth,termdepth,lineskip,lfdelay*, . .
.. *ffdelay,backspace,printwidth,printdepth*, . .
.. *termtype* {(R)}**

Specifies terminal and printer characteristics for current process. Any parameter may be left intact by entering a null (two consecutive commas). The *termtype* parameter may appear anywhere in the command. See also the DEFINE-TERMINAL and TERM-TYPE commands. The following is a list of term definitions:

termwidth	Number of print positions per line on terminal. Must be between 10 and 140
termdepth	Number of lines on terminal
lineskip	Number of blank lines at bottom of screen
lfdelay	Number of delay characters output after linefeed
ffdelay	Number of delay characters output after a top-of-form is executed
backspace	Decimal number of ASCII character to echo to the terminal when the backspace key is pressed
printwidth	Number of print positions per line on printer output
printdepth	Number of lines per page on printer output
termtype	Type of terminal (for BASIC and PROC cursor control) Terminal Types Supported: (Consult your vendor for additional types—this list was compiled from the types provided with the Pick XT and AT):

A	Adds 580	L	Lear Seagler
B	Ampex 210	M	Ampex 80
C	C Itoh VT52	P	Pertec 701
D	Datamedia	R	Adds Regent
E	Esprit	T	Televideo 910
G	IBM 3161/3163	V	ADDS Viewpoint
I	IBM Monochrome	W	Wyse 50

The option for TERM is:

R Redisplays TERM parameters after entry

A note about the *ffdelay* parameter: This parameter also determines when to eject a page (or clear the screen) between output pages. The following settings illustrate this concept:

ffdelay

Function

- 0 Suppresses page eject between output pages on both the terminal and the printer
- 1 Suppresses page eject on the terminal only
- 2 Normal mode. Any value greater than or equal to two will clear the screen (and eject a page on the printer) between each output page

>TERM-TYPE

Activates PICK/BASIC program to automatically set the terminal characteristics for the current line. The item read in to set the characteristics is

found in the DICT of the ACC file. The item-id is the 2-digit port number. The editor is used to maintain individual port characteristics. For example, to change the terminal characteristics for line 0, you could enter:

```
>ED DICT ACC 00<cr>
TOP
.L2<cr>
001 Memory Mapped Monitor
002 79,24,1,0,2,8,80,60,I
```

Compatibility: Pick AT/XT

>TEST-CURSOR

Activates program to test the various terminal capabilities such as clear screen, clear to end of line, “X,Y” addressing, reverse video, etc. See also the DEFINE-TERMINAL command.

Compatibility: Pick 2.2 and higher

>TIME

Displays current system time and date.

>UNLOCK-FRAME {*.*}fid

Removes previously locked frame from memory. See also the LOCK-FRAME command.

>UNPACK

Activates PICK/BASIC program to convert composite items back into the smaller, multiple items created with the PACK command. Pick Systems provides this program, but no support for it. See also the PACK command.

Compatibility: Pick 2.2 and higher

>VERIFY-SYSTEM {(O)}

Verifies the integrity of the Pick System software by comparing each attribute in the item called CHECK-SUM in the ERRMSG file against its corresponding virtual memory address. When differences occur, a message displays indicating that mismatches occur in one or more of the ABS frames. These can usually be corrected by performing an ABS restore from your sys-gen media. The option is:

O Generates new checksum item in ERRMSG file

>WHAT {*options*}

Outputs system status (normally produced by the WHERE command), configuration information, group and execution lock status (LIST-LOCKS), and the current status of the spooler (SP- STATUS). The options are:

'account name'	Outputs WHERE status for only the users of the specified account name
n	Outputs WHERE status for line <i>n</i> only
n-n	Outputs WHERE status for lines <i>n</i> through <i>n</i>
L	Suppresses display of lock status
P	Directs output to system printer, via spooler
S	Suppresses display of SP-STATUS
W	Suppresses display of WHERE command
Z	Displays WHERE status for all lines

>WHERE {(*options*)}

Outputs current execution information on processes currently logged on to system. The information displayed usually includes the port number, the PIB status, the process' PCB address, and the virtual return stack. The options are:

n	Displays information on line <i>n</i>
n-n	Displays information on lines <i>n</i> through <i>n</i>
'account name'	Outputs status for accounts logged onto specified account name
P	Routes output to system printer, via spooler
Z	Displays status of all lines

>WHICH

Provides current release level.

Compatibility: Pick AT/XT

>WHO {*line#*{-*line#*}}

>WHO {***}

>WHO '*acctname*'

Outputs account name being used on specified *linenumber*, range of line numbers, or the current account name when no line number is specified. The asterisk (*), displays the current account name for all lines, whether or not they are logged on. An account name may also be specified to determine the ports logged on to that account.

>X-REF

Assembler cross-reference verb used in XREF proc.

>XCS {*portnumber*}

Displays status of Extended Character Set for specified port number or current port if omitted. When XCS is enabled, the high-order bit is not stripped from input. This allows characters in the (decimal) range 128 to 239 to be used. See also the XCS-OFF and XCS-ON commands.

Compatibility: Pick 2.2 and higher

>XCS-OFF {*portnumber*}

Disables extended character set on specified port number or current port if omitted. See also the XCS and XCS-ON commands.

Compatibility: Pick 2.2 and higher

>XCS-ON {*portnumber*}

Enables extended character set on specified port number or current port if omitted. See also the XCS and XCS-OFF commands.

Compatibility: Pick 2.2 and higher

>XREF

Activates PROC to produce a cross-reference listing in the XSYM file of the most current CSYM file.

The System Debugger



The system debugger is used for assembly code debugging. It has the capability of accessing and changing data directly on the disk. A thorough familiarization with this section is recommended prior to using the debugger, as it can be *very* destructive.

SYMBOL FILE DEFINITIONS

The following are the symbol file definition:

PSYM Permanent symbol table
OSYM Object symbol table
TSYM Temporary symbol table

Note that these are the “standard” tables for “classic” Pick. These files are not always provided with each version of Pick and may actually go by different names. Usually, however there is a PSYM file provided with the Assembler account, which may have to be purchased or obtained separately.

THE SET-SYM COMMAND

The syntax for the SET-SYM command is:

>SET-SYM *filename* {(T)}

Sets symbol file pointer for use by system debugger. Filename is typically PSYM. The option is:

T Sets secondary file pointer to TSYM file

TERM CONVENTIONS AND DEFINITIONS

To address any disk location, usually three pieces of information are required: The *Data Format* Specification, the *Data Reference* Specification and the *Data Window* Specification. This may seem a little confusing at first. Several examples of how to put these pieces together are provided after the descriptions of the specifications.

THE DATA FORMAT SPECIFICATION

The DFS specifies the output display format mode of operation to use. It precedes the address of the data to access. The DFS, once set, will remain in the designated mode until explicitly changed.

All of the following are invalid with the use of the A or the L commands, and will generate error messages. The options are:

- C** Character format
- I** Integer format. Automatically generated by any reference to a symbol of the type: H, T, D, or F. A window specification of greater than six bytes will default to one byte
- X** Hexadecimal format

THE DATA REFERENCE SPECIFICATION

The DRS specifies either a direct or an indirect disk address.

➡➡➡ Address Specifications

Address specifications are represented by the character preceding the address, as follows:

- .address** Indicates the address is a *hexadecimal* number.
- ,address** Indicates the address is a *decimal* number.

The formats illustrated in this section use the default hexadecimal representations.

➤ Direct Reference

!fidaddr.dataaddr

!dataaddr

Where *fidaddr* (frame id address) is the logical frame, and *dataaddr* (data address) is the address of the unlinked data frame. Both of these parameters may be preceded by a period (.), indicating that the address is a hexadecimal value, or a comma (,), which indicates that the address is a decimal value.

!/fidaddr.dataaddr

Same as above, but the frame is assumed to be in linked format.

➤ Indirect References

The following are *implicit*:

!Rregaddr

Where register number address is the reference to data pointed to by a register number in the range 0–15.

!/symbolname

Where *symbolname* is the name of the symbol, as defined in either the PSYM or TSYM files. This will display the register number, displacement, format type, and window of the symbol.

Note that the symbol table must be set prior to using the indirect reference to a symbol name.

The following are *explicit*:

!*symbolname

References the data pointed to by the symbolic name of the register or storage register.

!Rregaddr.dataaddr

Applies the displacement, (*dataaddr*), to the location pointed to by *Rfidaddr* in order to obtain a storage register, with which the data may be addressed.

!*fidaddr.dataaddr

!*dataaddr

As above, but treats the specified location as a storage register. The displacement, *dataaddr*, is applied to the frame address to find the address of the storage register.

!symbolname**

!fidaddr.dataaddr**

!dataaddr**

References the storage at which the storage register points, with one condition: If the first byte of the medial storage register is a X'82', then the element is interpreted as a BASIC indirect string element and the storage register is taken from two bytes beyond this location.

If any of the data fields are invalid as storage registers, then the message ERR! will appear and some machines will croak.

THE DATA WINDOW SPECIFICATION

The DWS indicates the number of bytes to display and has the general form:

;window

Designates window as a tally for display, or a half-tally for the trace and Y-trace. This represents the number of bytes to display. The maximum is limited to the size of the frame being requested.

➤➤➤ Offset Specifications

The following are *explicit*:

;offset,window

Designates offset and window, where *offset* is a positive or negative tally indicating the offset from the DFS, and *window* is a positive number as above. This form works for traces, except in the case that the location is an indirect reference from a storage register whose location is specified by the form **!*fidaddr.dataaddr**.

The following are *implicit*:

;Offset

;Offset.window

Where offset and window are functionally the same as above, and the C (code) designates the number of fields. If the window specification is not included, then the implicit window derived from the field type is used, otherwise the specified window is used. The possible designators for the C parameter are as follows:

- B Field width = 1 bit
- C Field width = 1 byte
- D Field width = 4 bytes
- F Field width = 6 bytes
- H Field width = 1 byte
- R Field width = 8 bytes
- S Field width = 6 bytes
- T Field width = 2 bytes

;C

Where the C parameter is as above, but in this form, an offset of zero is used, along with the implied window and DFS.

In the specific case of bit (B) display, the general form:

;Boffset.window

Specifies bit display, starting at bit zero, the offset from the addressing base, for a width of window bits. Bits and bit fields may be traced with either trace. The displacement specified by a symbolically-addressed bit is in bits. Therefore, the form *fidaddr.dataaddr* will treat the *dataaddr* as a bit count in the direct-reference form.

Here are some examples of putting together the DFS, DRS and DWS:

!C4888, 12; 4<cr>

This reads : in character format, frame 4888, displaced (or offset) in 12 bytes, retrieving a window of 4 bytes.

!X.FEC.C;.A<cr>

This reads : in hexadecimal format, frame FEC, displaced in C bytes, retrieving a window of A bytes.

!C.FEC,12;4

This illustrates that different combinations of decimal and hexadecimal parameters can be specified.

DISPLAY PROMPTS AND SPECIAL FUNCTIONS

Once a valid address has been specified, the debugger accesses the requested location and displays its current contents. The cursor is positioned to the right of an = sign, indicating that the debugger is ready to either accept any of the following special functions or change the contents of the window.

The following commands are expected in response to the equal (=) sign that appears as the result of a legal debugger command.

. .=<cr>

(Carriage Return). Returns control to the command processor. Leaves window unchanged.

. .=<lf>

(Linefeed). Displays the next window of data, on the same line.

. .=<control>N

Displays the address of the next window and the next window on the next line.

. .=<control>P

Displays the address of the previous window and the previous window on the next line.

. .=*'string*

Replaces the characters specified in *string* at the beginning of the displayed window for the length of *string*, which may not exceed forty bytes. The string must terminate with a <cr>, <lf>, <control>N, or <control>P. This is used to change data displayed in “character” format (see the DRS).

. .=*decimal number*

Places the value of the decimal number in the displayed window, filling from the right, if the window is 1, 2, 4, or 6 bytes in length, and does not cross a frame boundary. The string must terminate with a <cr>, <lf>, <control>N, or <control>P.

. .=*hexadecimal string*

Places the value of the hexadecimal string in the displayed window, filling from the left. The string must contain an even number of hexadecimal characters, and may not exceed 38 hexadecimal characters. The string must

terminate with a <cr>, <lf>, <control>N, or <control>P. This is used to change data displayed in hexadecimal format (see the DRS).

. .=*binary string*

Binary String. The *binary string* is a sequence of 1's and 0's less than 40 characters in length. Places the value of binary string in the displayed window, starting with the first bit in the window. The string must terminate with a <cr>, <lf>, <control>N, or <control>P.

. .=0****

(Zero). Clears the window to null, if the type is not bit display. The string must terminate with a <cr>, <lf>, <control>N, or <control>P. Only functional on binary string.

. .=A****

Redisplays the address of the last window, along with the window.

. .=C****

. .=Cwindow****

.. =Coffset.window****

Changes the display type, window and offset if specified, and redisplay either the original field with the new type and/or window specification, or the resultant field if the offset is changed. The string must terminate with a <cr>, <lf>, <control>N, or <control>P.

SYSTEM DEBUGGER COMMANDS

!A{DRS}

Displays the current instruction location of the virtual code address. The DFS is meaningless to this command and will generate the error message, ILLGL SYM (illegal symbol).

!ADDD *decnum decnum*

Adds two decimal numbers and displays result in decimal.

!ADDX *hexnum hexnum*

Adds two hexadecimal numbers and displays result in hexadecimal.

!B{DRS}{.0}

Sets a breakpoint at specified data reference location, causing a break each time the address is encountered. The optional zero (0) indicates that every entry point in the frame is to be treated as a breakpoint, in which case, the *dataaddr* parameter is omitted from the DRS. The DRS may contain a maximum of two numeric parameters, hence no window specification is expected.

A plus (+) character is displayed for each breakpoint successfully entered into the table, until the table is full. The maximum number of breakpoints is four.

!C{DRS}

Activates Character display format. See section on DFS – Data Format Specifications.

!D

Displays table containing current breakpoints, traces, data breakpoints (y-trace tables), and frame replacement specifications.

!DB

Toggles the debugger availability flag. Must be executed from SYSPROG.

!DIVD *decnum decnum*

Divides two decimal numbers and displays result in decimal.

!DIVX *hexnum hexnum*

Divides two hexadecimal numbers and displays result in hexadecimal.

!DTX *decnum*

Converts decimal value to hexadecimal equivalent and displays result.

!DTX {*radix*} *decnum*

Converts decimal value to a specified *radix* (base) and displays result. Default radix = 16.

!E{*numberlines*}

Specifies number of instructions to execute prior to returning to debugger command level. The *numberlines* parameter must be in the range, (0–250).

The E command, followed by a <cr>, disables previous iteration counter setting.

!END

!end

Terminates debugger process and returns control to TCL. A change was made to this command in release 2.2. Formerly, if the system debugger was entered from the PICK/BASIC debugger, an END would return control to the PICK/BASIC debugger. Now, it returns directly to TCL.

!Foldfidaddr.newfidaddr

Frame replacement function. Replaces entries in *oldfidaddr* as entries in *newfidaddr*. Both parameters may be either decimal or hexadecimal numbers. See section on Address Specifications under DRS — Data Reference Specifications.

!G(DRS)

Go function. Transfers control to address specified in the DRS, with the exception of not allowing window specifications. The G command, followed by a <cr>, continues execution, if possible, from the current virtual address location.

!H

Toggles the ECHO flag.

!I(DRS)

Activates Integer display format. See section on DFS — Data Format Specifications. The I command, followed by a <cr>, will display subsequent output in Integer format.

!K(DRS)

Kills breakpoint previously specified with the B command. See B command for limitations of breakpoint addresses. The K command, followed by a <cr>, kills all current breakpoints in the table.

A minus (–) character is displayed for each breakpoint entry successfully removed from the table.

!L(DRS)

Displays the link fields of the frame implied in the DRS. A DFS may not be used with this command and will generate an error message, if used.

!M

Toggles modal trace function on or off. The modal trace function, when on, enters the debugger each time a frame boundary is crossed.

!ME{*line#*}

Assigns all PCB and symbolic reference data specifications to the specified line number. The ME command, followed by a <cr>, resets the pointer to the current virtual process' PCB.

!MULD *decnum decnum*

Multiplies two decimal numbers and displays result in decimal.

!MULX *hexnum hexnum*

Multiplies two hexadecimal numbers and displays result in hexadecimal.

!N{*tally reference counter*}

Prints the instruction address and other characteristics for the number of times specified in the tally reference counter prior to return to the debugger command level, unless a real error is encountered, in which case entry to the command level is automatic. The N command, followed by a <cr>, cancels this function, such that all breaks will re-enter the debugger command level.

!OFF

Terminates process, logs process off of system, and returns control to logon message. A change was made to this command in release 2.2. Formerly, if the system debugger were entered from the PICK/BASIC debugger, an OFF would return control to the PICK/BASIC debugger. Now, it logs off.

!P

Toggles the LISTFLG function, either enabling or disabling output display.

!R{*regaddr*}

Indicates indirect reference to specified register number address. See the section on DFS — Data Format Specifications.

!SUBD *decnum decnum*

Subtracts two decimal numbers and displays result in decimal.

!SUBX *hexnum hexnum*

Subtracts two hexadecimal numbers and displays result in hexadecimal.

!T{DRS}

Sets a trace table entry, indicating to the debugger to display the specified data element, along with its address, on each break. If the DRS is omitted, the T command toggles the trace function on or off. The system will display the appropriate condition.

A plus (+) character is displayed for each trace table entry successfully entered into the table, until the table is full. The maximum number of trace table entries is four.

!TIME

Displays the current system time and date.

!U{DRS}

Removes trace table entries previously specified with the T command.

A minus (–) character is displayed for each trace table entry successfully removed from the table.

The U command, followed by a <cr>, clears all commands previously specified with the T command.

!X{DRS}

Activates Hexadecimal display format. See section on DFS – Data Format Specifications. The X command, followed by a <cr>, displays subsequent output in Hexadecimal format.

!XTD *hexnum*

Converts hexadecimal number to its decimal equivalent and displays result.

!XTD {*radix*} *hexnum*

Converts *radix* (base) to its decimal equivalent and displays result. Default radix = 16.

!Y{DRS}

Sets a y-trace table entry indicating to display the specified data element, along with its address, each time the data element changes. If the DRS is omitted, the Y command toggles the function on or off.

The current value of the data, stored in aligned words, is kept with the address data, so that the table element size will change with varying sizes of data.

A plus (+) character is displayed for each entry successfully entered into the table, until the y-trace table is full. The maximum number of entries is four.


The y command slows down execution speed drastically.

!Z{DRS}

Cancels data break commands set by previous Y command. The Z command, when followed by a <cr>, cancels all data breaks set by previous Y command.

A minus (–) character is displayed for each entry successfully removed from the y-trace table.

PICK/BASIC DEBUGGER



The PICK/BASIC Interactive debugger is used for tracing execution locations and variables in PICK/BASIC programs. At the time of this writing, no SMA standards documentation existed for the PICK/BASIC debugger.

➤ Symbol Definitions

The symbol definitions are automatically defined during the compile process, and concatenated to the object code pointer in the dictionary level of the program file.

ACTIVATING THE DEBUGGER

There are several ways to enter the debugger. Some of them are voluntary such as: 1) Pressing the “break” key while a PICK/BASIC program is executing; 2) Running the program with a “D” option. For example:

```
>RUN BP ENTER.CUSTOMER (D)<cr>
```

or, if the program is cataloged, providing a “D” option after the program name at TCL:

```
>ENTER.CUSTOMER (D)<cr>
```

3) Placing a “DEBUG” statement in the source code of the program at the point where you want to enter the debugger.

The other means of entering the debugger is, of course, involuntarily. This occurs during a “fatal” error condition, like trying to write to a diskette that is still in its jacket or referencing a file that hasn’t been opened. Refer to the PICK/BASIC Error Message section of this book for clues to the causes and cures of most of the fatal error conditions. The debugger is usually gracious enough to give you an error message number and a vague clue of why the program died.

➤ Prompt Character

The prompt character, asterisk (*), appears in the leftmost column of the terminal display screen, indicating that the debugger is ready to accept any legal command.

➤ Operators

The operators listed below perform a logical comparison function and are used in conjunction with the “B” command for setting breakpoint conditions.

=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
#	Not equal to

➤ Referencing Variables

***/variable**
***/variable(*n*)**
***/variable(*n*,*n*)**
/

Displays current value of specified string or array variable and allows operator to change current value. If the literal, *, is entered, then the values of all variables defined in the program are displayed.

****/variable((dimension1,dimension2))***

Displays current value of array element defined in specified variable. The variable name, followed by a <cr>, displays all elements in the array, when more than one is defined, until the last element is displayed or the break key is pressed.

➤ Zone Output Specification

****[(leftmargin,rightmargin)]***

****[***

Sets left and right margins for output zone limits of debugger display. The [command, followed by a <cr>, removes zone limits.

PICK/BASIC DEBUGGER COMMANDS

****?***

****\$***

Displays current program name, execution line number, and object code verification status.

****Bvariable operator variable***

****Bvariable operator "literal"***

****B\$operator linenumber***

Sets a breakpoint in the trace table, indicating that control shall pass to the debugger when the test condition defined in the command expression is met. The dollar sign (\$) specifies program line numbers for execution/breaks. A plus (+) character is displayed for each breakpoint successfully entered into the table until the table is full. The maximum number of breakpoints is four.

Note that the spaces between the arguments in the above syntax are simply there for readability — they are not allowed when composing breakpoints, as shown in the following examples:

****BAMOUNT.DUE>***

Specifies for the debugger to be entered when "AMOUNT.DUE" is greater than 0.

****B\$=117***

Specifies re-entry into the debugger before execution of line 117.

***D**

Displays the contents of the Break and Trace tables.

***DE**

***DEBUG**

Transfers control to the system debugger. See the System Debugger section for available commands.

***E{*numberlines*}**

Specifies the number of instructions to execute prior to returning to debugger command level. The E command, followed by a <cr>, disables previous iteration counter setting.

***END**

Terminates program execution and returns control to TCL.

***G{*linenumber*}**

“Go to” function. Transfers control to a specific program (source) line number. The G command, followed by a <cr>, resumes program execution from the current program line number.

Note that if the program was compiled without EOL (End-Of-Line) characters, then the only number allowable for the “GOTO” is one (1).

***K{*breakpoint-number*}**

Kills a breakpoint previously set with the B command, and removes the entry from the breakpoint table. The K command, followed by a <cr>, removes all breakpoint entries from the table.

***L{*startingline-numberlines*}**

***L{*numberlines*}**

***L{*}**

***L**

Displays source code program lines, beginning from specified starting line number, or from current position, if no starting line number is specified. See also the Z command. The L command, followed by an * (asterisk), displays the entire program. The L command, followed by a <cr>, displays the current program source code line.

***LP**

Toggles the line printer bit, either directing debugger output to the terminal screen or the spooler.

***N(*numbertimes*)**

Instructs the debugger to ignore breakpoints for the next number of times encountered. The N command, followed by a <cr>, resets the bypass, and breakpoints are processed at each occurrence.

***OFF**

Terminates program execution, logs process off system, and returns control to the logon message.

***P**

Toggles the LISTFLG function, either enabling or disabling output display.

***PC**

Closes the currently open spooler file entry, releasing control to spooler.

***R**

Removes the top return stack address of local subroutine from stack, causing the program to return from current subroutine as though a RETURN statement was encountered.

***S**

Displays the contents of the subroutine stack.

***T(*variable*)**

Sets a trace table entry, indicating to the debugger to display the specified data element, along with the contents of the break and trace tables, on each break. A plus (+) character is displayed for each trace table entry successfully entered into the table, until the table is full. The maximum number of trace table entries is six. The T command, followed by a <cr>, toggles the trace function on or off.

***U{*trace table entry*}**

Removes trace table entries previously specified with the T command. A minus (—) character is displayed for each trace table entry successfully removed from the table. The U command, followed by a <cr>, clears all trace table entries previously specified with the T command.

***V**

Verifies PICK/BASIC object code. No longer serves any useful function.

Z {DICT} *filename itemname

***Z<cr>**

FILE/PROG NAME?*filename itemname*

Specifies that the debugger should use the symbol table defined for the program referenced in itemname, in the specified filename. The Z command, followed by a <cr>, prompts for the filename and itemname.

The Spooler



Most spooler commands allow options. Sometimes, these options have numeric arguments. In many cases, valid combinations of options are available, but need to be made clear to the options interpreter. Generally, it's a good idea to separate options with blanks when options contain numbers, as in the following example:

```
>SP-ASSIGN HS F1 3
```

Assigns for hold file output, to *formqueue* 1, with 3 copies. Notice that spooler options do not have to be enclosed in parentheses like options with other TCL commands. There is, however, an important rule to keep in mind:

IF NUMERIC OPTIONS ARE WITHIN PARENTHESES,
PARAMETERS OUTSIDE OF THE PARENTHESES WILL BE
IGNORED!!

SPOOLER COMMANDS

```
>:STARTSPOOLER {(option(s))}
```

Initializes spooler and returns to normal operation. The *options* are:

- n Starts spooler process on specified port number. (Used primarily by developers to test the spooler code).
- C Clears all but permanent file control blocks (hold files). Does not return space to overflow table.
- D Used in conjunction with the “n” option above to enable the debugger on the line.
- I Initializes and DESTROYS all spooler queue entries. Does not return space to overflow table.
- L Links workspace for inactive lines.

>LISTABS (*Options*)

Outputs current spooler assignment parameters for every port on the system. See the SP-ASSIGN command for interpretation of the displayed status codes.

>LISTPEQS (*Options*)

Outputs list of spooled entries. The *options* are:

- 'acctname' Displays spooler entries created by specified account name
- n Integer number, in the range (0–600), indicating the spooler entry to display
- n-n Specifies beginning and ending spooler entries to display
- A Outputs the status of all reports created from current account
- C Outputs count of spooled entries and total storage (frames) used
- E Outputs print file absolute location
- F Outputs status of jobs queued for output in queue order (0–125)
- H Outputs hold files only
- L Outputs active and deleted reports
- P Directs output to system printer, via spooler

The following are the statuscodes:

- A Available
- C Closed
- G Align
- H Hold file
- I Immediate
- L Locked
- N No close
- O Currently being output
- P Printer
- R Requested
- S Spooled
- T Tape
- X Aborted

>LISTPTR (*Options*)

Outputs the printer control block status for all printers. The *options* are:

- n{-n) Specifies beginning and ending printer numbers
- N Activates NOPAGE function on output to terminal
- P Directs output to system printer, via spooler

>SP-ASSIGN (*Options*)

Sets spooler output assignment for current process. Note that an SP-ASSIGN command with NO options resets the assignment back to the default state. This MUST be done prior to using the SP-EDIT command on hold files when attempting to de-spool. The *options* are:

- n Integer number in the range (1 – 125) indicating the number of copies (Default = one)
- ? Outputs resulting setting, or current setting when entered alone
- C Activates choke function. *Choke* means that the spooler will require no more than 20 frames to print the job, and will recycle the 20 frames. Must be used with an I option, and may not be used with H option
- Fn Directs output to *formqueue n*, where *n* is an integer number in the range (0 – 125)
- H Directs output to hold file(s)
- I Instant print. Begins output (assuming device is ready) as soon as the first frame of output is completed
- M Suppresses “Entry n” message
- O Indicates to leave file open upon completion of process. This allows placing multiple print jobs in one physical spooler entry. See also the SP-OPEN and SP-CLOSE command. To close the entry, use the SP-ASSIGN command without options
- Rn Initiates the print file, “n”, where “n” is an integer number in the range (0 – 125) for use with BASIC “PRINT ON” statements
- S Suppresses printing on system printer. Normally used with H option
- T Attaches tape unit as output device

Note that when using the *n*, *Fn*, *Rn* and *Qn* options above must be separated by a blank from any other options present in the command line. Here are some examples of the SP-ASSIGN command:

>SP-ASSIGN HS?

>SP-ASSIGN 13 F2 ?

>SP-CLOSE

Closes the currently open spooler entry. See also the SP-OPEN command. Entries are automatically closed if a LOGTO, OFF, or SP-ASSIGN (without an R option) is issued.

>SP-EDIT {*Options*}

Accesses spooler entries for processing or deletion. When options are omitted, allows access to all entries generated from current account. The options are:

'acctname'	Accesses entries generated from the specified account
n{-n}	Integer number, in the range (0–600), of spooler entry, or beginning and ending hold spooler entries
D	Delete{s} hold file{s}
F{b{-e}}	Integer number, in the range (0–125), specifying the beginning and ending form queue{s} to access
H	Suppresses tape label when directing output to tape
L	Activates “look” function, and displays first 500 bytes of spooler entries
MD	Manager (also called “magic”) priority Delete{s} spooler entries
MS	Manager priority that spool{s} spooler entries
N	Activates NOPAGE function when spooling to terminal
O	When used with the L option, allows looking at spooler entries already being output
P	Directs hold file to currently assigned printer
R	Changes output parameter settings to currently assigned parameter{s}
T{W}	Directs output to tape. W indicates to wait for tape to be readied, if not already
U	Accesses all spooled entries generated from current process
V	Converts hold file to data file, without trimming blanks and blank lines

Here are some examples of the SP-EDIT command:

```
>SP-EDIT MSP
>SP-EDIT MSP1–5
>SP-EDIT MD1–10
>SP-EDIT MUD
```

>SP-KILL {*Options*}

Stops output of currently active spooler entry on specified printer. Here are the options:

n{-n}	Integer number, in the range (0–19) printer to stop
A	Stops spooler entry or entries created from current account
B	All spooler entries
Dn{-n}	Deletes printer{s}. The beginning and ending printer number{s}

- must be in the range (0–19). Note: the maximum number of printers may vary between various PICK-class systems
- F{b(-e)} Returns spooler entries to available hold file status. The beginning and ending parameter{s} must be in the range (0–600)
- N Suppresses abort message on output when directed to a printer
- O Unlinks spooler entry already being output

Here are some examples:

```
>SP-KILL 1
>SP-KILL OF7
```

>SP-OPEN

Specifies that spooler is to leave the spooler entry open upon completion of process. See also SP-CLOSE and SP-ASSIGN O option.

>SP-STATUS {*Options*}

Displays status of all spooler-controlled devices. The *options* are:

- n{–n} Integer number indicating the beginning and ending printer number{s} to display
- B Outputs status of all printers
- P Directs output to system printer, via spooler.

>SP-TAPEOUT {*Options*}

Retrieves reports previously spooled to magnetic tape, and outputs to current device assignment. The *options* are:

- A Converts EBCDIC format to ASCII format
- L Allows output of tapes created with each line being treated as a tape record
- U Converts all lower case characters to upper case

>STARTPTR *printer#*,*formqueue#*,*#ejectpages*, *printer specification*,{A}{{S{Xlinecount}}{(D)}}

Activates and initializes parallel or serial printer. The printer specification is composed of two parameters; the type (P or S), and the address or location of the device. The letter A may follow the printer specification. When used, it allows for forms alignment on the newly started printer. The printer types are:

- P Parallel. This requires special hardware. Consult your vendor to find out if more than one parallel printer is supported.
- S Serial. Can go in any serial I/O port.

Here are some examples of printer specifications:

P0 Parallel printer 0
S12 Serial printer on port 12

The *options* are:

D Enables debugger on line. (Used primarily by developers to test spooler code.)
S Suppresses first form feed on serial printer
Xn Specifies the number of lines per page for serial printers

Here are some examples of the STARTPTR command:

>STARTPTR 0,0,0,P0
>STARTPTR 1,1,0,S12

>STOPPTR (*Options*)

Stops the specified printer at the end of the current print job. The default printer is printer number zero (0). Here are the *options*:

n Integer number of printer to stop, in the range (0–19)
B Stops all printers

System Error Messages



Many error messages that still remain in the ERRMSG file are no longer used, or are too obscure to document. They are still included in the unlikely case that they will ever be needed.

Error messages have their own unique “language”, and are handled by a special program contained in the operating system used to format their output according to the various “instructions” contained in the message, much the same as in PROC.

ERROR MESSAGE FUNCTIONS

The following chart illustrates the various instructions available to the error message processor and explains what they mean:

<i>Function</i>	<i>Description</i>
A	Outputs argument at current screen coordinate
A(n)	Outputs argument, left-justified, in a field of “n” blanks
D	Outputs system date at current screen coordinate
E	Outputs error message item-id, enclosed in brackets

<i>Function</i>	<i>Description</i>
Hstring{+}	Outputs literal string at current screen coordinate. The optional + sign is used to hold the cursor position, typically for input
L	Issues a linefeed
L(n)	Issues <i>n</i> linefeeds
R(n)	Outputs the next argument, right-justified, in a field of <i>n</i> blanks
S(n)	Positions the cursor to column (position) <i>n</i> on the current row (line)
T	Outputs system time at current screen coordinate
X	Ignores the next argument in the argument list

TESTING ERROR MESSAGES

The PRINT-ERR verb can be used to test error messages. It has the general format

```
>PRINT-ERR filename item-id
```

The filename is usually ERRMSG.

THE SEQUENCE OF ERROR MESSAGES

There are several important error messages to know about. The first is an item called “LOGON”, that resides in the SYSTEM file. It is used to display the “logon” message, that is, the message that displays on a terminal when the terminal is logged off.

When you log on, the next error message that is activated is an item also called “LOGON”, which resides in the ERRMSG file. After that, error message “335” is displayed. That’s the one that “welcomes you” to the system and optionally tells you the time, date and release number.

When you log off, error message “336” is displayed.

THE STANDARD PICK SYSTEM ERROR MESSAGES

1 PASSWORDS CAN ONLY BE CHANGED ON SYSPROG

2 Uneven number of delimiters (‘ “ \).

There must be an even number of single quotes (‘), double quotes (“) and/or backslashes (\) in any TCL command.

3 Verb?

Unrecognizable command issued at TCL. Most often cause is the fact that the verb is not defined in the current accounts' Master Dictionary.

5 The word "item" is illegal.

Means that the item referred to as "item" has an "X" in attribute one in the dictionary, defining it as a "protected" attribute.

7 A value must follow the HEADING, FOOTING, tag or GRAND-TOTAL connective.

Each of these modifiers must be followed by a text string enclosed in double quotes.

8 A window specification string must follow the window connective.

9 System D-pointer missing.

10 File name missing

The specified file name either has an incorrect file definition item, or points to a file that can not be located.

11 Frame locked at location x'location'

Displayed when using the LOCK-FRAME command to permanently corelock a particular frame.

13 Data level descriptor (file-name in dictionary) is missing.

Means that the D-pointer to the data section of a file has been changed or deleted from the dictionary.

15 The file-name is preceded by an illegal connective.

Indicates that a connective (such as AND, BY, or BREAK-ON) preceded the file name. Connectives are followed by attribute names, not filenames.

17 WITHIN is valid only in COUNT, LIST, SUM or STAT statements.

18 The last word may not be a connective.

Typically caused by ending an ACCESS sentence with a connective, such as AND.

19 A value without an attribute name is illegal.

Typical cause is a value enclosed in double quotes in an ACCESS sentence without a WITH clause or enclosing itemnames with double instead of single quotes.

20 Error in the USING syntax.

21 Meaningless item-id in statement

22 TO before item-id valid only in a CHANGE statement

24 The word “item” cannot be identified

The specified Attribute-Defining-Item either is improperly formatted, or does not exist as entered.

25 WITH may not immediately precede a value.

Means that the Attribute-Defining-Item is missing between WITH and value. Form of selection criteria is: WITH *attributename* {*operator*} “*value*”.

**26 Attribute values may not both precede and follow an attribute name.
Form of selection criteria is: WITH *attributename* {*operator*} “*value*”.**

27 The word WITH is missing

29 At least one item-id must be specified for a WITHIN-type statement.

30 Error in M/DICT definition of the verb.

Invalid verb definition. Check verb definition (in MD) for non-hexadecimal characters in attribute two, and compare the verb definition with the same item in SYSPROG’s MD or the NEWAC file.

53 Item list required for WITHIN

71 An illegal connective modifies the word ‘word’

An Attribute-Defining-Item has been preceded by an illegal connective, such as an “AND” connective.

72 The value “value” is meaningless.

Returned upon unsuccessful execution of the user-exit Un1AD, where “n” is the entry point.

79 The number of separate AND clause sets cannot exceed 9.

There is a maximum of 9 AND clauses in an ACCESS sentence.

80 A system error has occurred in mode: modename. This may be due to sort-key(s) preceding selection criteria.

This is normally caused by using an “AND” connective without a “WITH” connective.

82 Your system privilege level is not sufficient for this statement.

An operation was attempted that is prohibited by definition of the system privilege level. Check account name in SYSTEM file. Attribute 8 may either be: SYS0 (lowest priority), SYS1, or SYS2 (highest priority).

91 End tape check – n file(s)

92 End of recorded data – n file(s)

Displayed at the end of a T-EOD command.

93 Attach the tape unit.

A tape operation command must be preceded by a T-ATT (tape attach) command. Once attached, it need not be re-attached unless a different blocksize is required.

94 End of file

Displayed when an EOF (End Of File) mark is encountered during a T-FWD (tape forward) or T-BCK (tape backward) command.

95 Tape attached to line ‘n’.

Tape must be detached from specified line number with the T-DET command prior to any operations.

96 BOT

Beginning-of-tape. Displayed when T-BCK (tape back) command was issued on the first file on the tape.

97 End of tape

Displayed when any tape moving command reaches the end of tape.

98 Parity error!

Means big trouble. Hopefully, this does not display during a full restore. It normally means that the tape has media problems and is unreadable. Parity errors are a good reason to check your FILE-SAVE tape immediately after it is created, using a “dummy” restore. A “dummy” restore is done using the “SEL-RESTORE” command. When asked for the account name, use a word that is *not* a valid account name (like Rumpelstiltskin) and a file name that is also *not* valid (like Skywalker, in case there is a Rumpelstiltskin account you were not aware of).

100 ‘item’ is not ‘whatever’

118 The form ‘with attribute and attribute’ is undefined

158 An illegal connective of the form “value” modifies “value”.

163 Attribute for SORT, BREAK-ON or TOTAL connective missing

164 Total or control break connective not succeeded by attribute definition

Means that a TOTAL or BREAK-ON modifier was not followed by a valid attribute name.

166 The A correlative attribute name ‘item’ is illegal.

167 Missing terminal quote (“/”) in A correlative: ‘item’.

Look at the specified attribute name in attribute 8 for the A correlative. Make sure that there is an even number of quotes.

168 Illegal A correlative.: ‘n’.

Indicates that the A correlative has an unrecognizable syntax. Could be a million or so different reasons. Look at the Correlative section of the ACCESS chapter for syntax on the A correlative.

169 Missing left paren in A correlative: “n”

Look at specified attribute name in attribute 8 for the A correlative. Make sure that each set of parentheses is “opened” and “closed”.

170 Missing right paren in A correlative: “n”

See error message 169.

173 Missing semi-colon in A correlative.

192 This is an erroneous object string.

193 Basic object code cannot be copied to an item.

194 Don't copy to the same item in the same file

This occurs when the following syntax is used in a COPY process:

```
>COPY old-filename *  
TO:new-filename
```

Simply put, COPY copies the first item in *old-filename* to an ITEM called *new-filename*, and then tries to do the same for the next item in *old-filename*.

To correct this, precede *new-filename* with a left-parenthesis, as follows:

```
>COPY old-filename *  
TO:(new-filename)
```

196 'item' is too large to be an item.

197 This get-list or save-list specification is incorrect

198 The list pointed to by pointer item 'item' is defective.

199 Insufficient work space for item 'item'.

Typically displayed when a very large PICK/BASIC is executed. The best way to correct is to break up the program into smaller, more manageable subroutines.

**200 200 'POINTER-FILE' IS NOT A POINTER TYPE (DCODE = 'DC')
FILE**

201 'filename' is not a file name

Indicates that the file name specified in a TCL command does not exist, or is improperly defined in the Master Dictionary of the current account.

202 'item-id' not on file.

Normally appears in conjunction with a GET-LIST command followed by an ACCESS command. Typically indicates that an item has been deleted from the file since the list was saved.

203 Item name?

Indicates the *itemname* or *itemlist* specification is missing from the TCL-II command.

204 File definition 'filename' is missing.

File (D—) pointer missing from M/DICT.

205 No statements to be assembled.

An assembly process was attempted on an item that either does not exist, or has no lines to assemble.

208 Error in item-id list

210 File 'filename' is access protected.

An operation was attempted on a file that has access code locks present. See System Administrator for gaining access to file. Literally means that the access and update codes do not match the user logon corresponding entry in the SYSTEM file. Ensure that the correct account is being used.

211 Mode 'modename' no assembled code can be found.

Displayed upon attempting to MLOAD or MVERIFY an assembler object frame that contains no object code.

212 Mode 'modename' has no frame statement

The first line of an assembler program must contain a FRAME statement, otherwise it is incapable of being verified or loading into the ABS area.

213 Mode 'modename' location counter error at line no. "n"

Output from MLOAD verb. Indicates that continuity of object code location counter in assembled assembly code is corrupted or has an illegal value. Reassemble mode or restore good item from FILE-SAVE tape.

214 Mode 'modename' overflows frame frame# at line # line#

215 Mode 'modename' hex error at line no. line#

Output from MLOAD verb. Indicates that there is illegal object code in item. Reassemble or restore good item from FILE-SAVE tape.

- 216 Mode 'modename' loaded;frame = fid size = size cksum = check-sum**
- Displayed upon the successful loading of an assembler (object) frame into the ABS area.
- 217 Mode 'modename' verified; frame = frame; size = size; cksum = checksum**
- Displayed upon execution of the MVERIFY command, used to compare the contents of the frame in the ABS area with the actual code.
- 218 Mode 'modename' frame = frame has n mismatches**
- Displayed when an MVERIFY operation (the verb used in VERIFY-SYSTEM) has found that the object code in the file item does not exactly match the code loaded in the corresponding frame in the ABS area. This may be corrected by reloading the mode, or all ABS frames.
- 219 Illegal command:command**
- Displayed when the RUNOFF process interprets an illegal command.
- 220 'itemname' exited**
- Used by Editor.
- 221 'itemname' filed.**
- Used by Editor.
- 222 'itemname' deleted.**
- Used by Editor.
- 223 'itemname' exists on file.**
- 224 'item' is a pointer item. It may not be loaded into a non-pointer file.**
- 225 TSYM must be dict & data mod/sep 1,1 37,1 sorry for the inconvenience.**
- The TSYM (Temporary symbol table) file must be created with both a dictionary and a data section.
- 226 Tape format error**
- See error message 98. This is even more insidious than parity errors, because this kind of problem does *not* get detected during a T-CHK. It literally means that the tape is not "logically" constructed properly. These

types of errors can usually be detected by running a “fake” or “dummy” restore after a FILE-SAVE.

234 Item size exceeds 32,000 bytes.

Source unknown. Normally, an abort will occur if an item exceeds the 32K limit. The editor has its own message for this problem. Let us know if you find what causes this.

235 Attempt to write into update protected file!

239 ‘filename’ is not a pointer type (dcode = ‘dc’) file

240 a “SELECT” or “SSELECT” must be used preceding a ‘save-list’ statement!

Displayed when a SAVE-LIST command is not preceded by a GET-LIST, SELECT, SSELECT or QSELECT command.

241 Successful compile!! n frames used.

Displayed when a PICK/BASIC program miraculously survives the compile process. Note that this message has absolutely nothing to do with the program running successfully.

242 ‘itemname’ decataloged.

Displayed when removing the object code for a PICK/BASIC program with the DECATALOG command.

243 list saved — n frames used.

Displayed after saving a list with the SAVE-LIST command.

244 ‘item’ cataloged!

Displayed when using the CATALOG command on a PICK/BASIC program.

245 ‘item’ list deleted

246 ‘item’ is no longer a list or no longer on file

247 ‘item’ is basic object code

Usually displayed when an operation such as EDIT is performed on the DICT (dictionary) level of a program file, where the object pointers are

stored. Not generally a good idea; that's why it is prevented. If you really want to play around with the object code, use the system debugger and the DUMP command.

260 Invalid batch lock command: command

265 PROC stack overflow

267 PROC transfer to 'procname' cannot be completed.

Displayed when a PROC attempts to "link" another PROC which it can not find.

268 The destination of the PROC "GO" statement: GO n, cannot be found

This normally means that there is a statement in the PROC telling the PROC to "GOTO" a specific statement label, which can not be found.

269 Input to proc buffer cannot exceed 298 characters

270 Format error in the proc statement: PROC statement

Displayed when an illegal statement is encountered in a PROC. Another possible cause is when a PROC gets updated (edited) while it is being run. Never, ever, change PROCs while they're running!

272 A value exists for the attribute referenced by the element : element

273 Error in column-number/field-width or format specification at: location

274 Unrecognizable batch-string element: element

275 Y or F sub-element error at batch-string element : element

276 D-2 update without d-1 being specified, at batch-string element : element

277 J element missing at batch-string element : element

278 Error in processing secondary batch-string element : element

279 Incorrect scaling factor in f* batch-string element : element

280 File-definition batch element error at : element

281 D1 must have Y11 storage correlative error at : location

282 Invalid parameter for SELECT list

289	TERMINAL	PRINTER
	PAGE WIDTH:	termwidth printerwidth
	PAGE DEPTH:	termdepth printerdepth
	LINE SKIP:	lineskip
	LF DELAY:	lfdelay
	FF DELAY:	ffdelay
	BACKSPACE:	backspace
	TERM TYPE:	termtype

See the TERM and the SET-TERM commands in the TCL section.

290 The range of the parameter “parameter” is not acceptable.

This is used by two processes : First, the TERM and SET-TERM commands, to indicate arguments that are too high. Second, this is used by the LIST-LABEL and SORT-LABEL commands when the total width of output for the labels exceeds the device width to which the output is being directed. Simply fix the TERM characteristics and try again.

293 Total number of contiguous frames : n

Displayed in the POVf command.

298 Format error in specifications.

316 Numeric parameter missing

331 The account file is missing.

Indicates that ACC (Accounting History File) has either : a) been deleted, b) is improperly defined in the SYSPROG account or SYSTEM file, c) or the Q-pointer to it in the current account MD is missing or improperly constructed. This sometimes occurs when logging on to a system while a restore is in progress. If a restore is *not* in progress, this may be corrected by performing a SEL-RESTORE of the ACC file from a FILE-SAVE, or by correcting the Q-pointer.

335 The PICK System for the IBM PC-AT Ver 2.n

See “The Sequence of Error Messages” at the beginning of this section.

336 < Logged off at time on date>

See “The Sequence of Error Messages” at the beginning of this section.

337 User is not logged on.

This displays when sending a message to a user who is not currently logged on.

338 Account file statistics were not updated due to either:

1. Insufficient work-space to contain the account file item, or
2. System dictionary changed while you were logged on.

Indicates that the ACC item is too large for available workspace. As a procedural recommendation, periodically clear the DATA section of the ACC file, but *not* while anyone is logged on. Additionally, if accounting statistics are not required, change attribute 9 in the SYSTEM file from “U” (Update ACC) to “L”, and avoid altering the SYSTEM file while users are logged on.

340 < Connect time= n mins.; cpu= n units; lptr pages= n>

341 PC-AT System verified.

342 *** PC-AT RELEASE 1.0 – SYSTEM DOES NOT VERIFY! ***

390 ‘frame’ LOADED.

391 ‘frame’ VERIFIED.

398 The file D-pointer requires a V(ertical) correlative for this statement.

The file D-pointer must contain a V code in attribute 8 to process ACCESS sentences containing the WITHIN connective.

399 The maximum of 20 levels for a WITHIN-type statement has been exceeded.

401 No items present.

Possibly the second-most popular error message ever displayed, right after error message 3 (Verb?). This means that the attempted selection criteria did not allow any items to be selected for processing. Also happens on empty files.

403 End of list

404 n items selected.

Displayed upon the successful completion of a SELECT, SSELECT, GET-LIST or QSELECT.

405 n items listed.

Displayed at the end of most ACCESS reports unless otherwise suppressed or indicated.

**406 item count= n, byte count= n, avg. bytes/item= n avg. items/group= n
std. deviation= n, avg. bytes/group= n.**

This is the summary line of HASH-TEST and ISTAT.

407 n items counted

408 one item counted

410 A synonym (q-type) file cannot be specified in this statement.

This means that an attempt was made to issue a DELETE-FILE or CLEAR-FILE command on a file designated as a Q-pointer. Fortunately, this is not allowed.

411 'DICT' or 'DATA' must be specified in a CLEAR-FILE statement.

This means that a CLEAR-FILE command did not indicate which level, either DICT or DATA to clear.

412 Insufficient disk space available for the file.

This displays when there is not a large enough block of contiguous disk space to fulfill the modulo request in the CREATE-FILE command. If the overflow table is heavily fragmented, consider doing a full restore to get all available overflow back into one contiguous block. Another alternative is buying another disk drive.

413 The file name already exists in the master dictionary.

This means that there is an item in the MD of the current account with the same item-id as the one requested in the CREATE-FILE command that just croaked.

414 Illegal or missing modifier used in defining the file area(s).

This indicates an error in the syntax of the CREATE-FILE command when creating a new file. See the CREATE-FILE command in the TCL section.

415 'item' exists on file.

Displayed by the COPY process, when the O (overwrite) option is *not* in effect, indicating that a duplicate item-id existed on the destination file, preventing the copy from taking place.

416 Range error in modulo or separation parameter.

This typically displays when one or both of the modulos are omitted from the CREATE-FILE command. See the CREATE-FILE command in the TCL section.

417 File 'filename' created; base = base, modulo = modulo , separ = sep (1).

Displayed upon the successful completion of the CREATE-FILE command.

418 File definition item 'filename' not copied

Displayed by the COPY process when a D-pointer is encountered. These are NEVER copied.

419 The specified file cannot be cleared or deleted!

This displays when someone foolishly tries to clear the SYSTEM file or their MD.

420 Dictionary file deletion cannot be done without deletion of data-section(s) first.

This means that an attempt was made to delete the dictionary level of a file without first deleting the DATA level.

421 Statistics of item : total = n average = n count = n

Displayed by the STAT command.

422 Byte statistics for : item total = n average = n items = n cksum = n bits= n

Displayed by the CHECK-SUM command.

423 Total of attribute = n

Displayed by the SUM command.

425 INVALID FRAME-ID REQUEST

This means that a frame-id less than 0 or greater than *maxfid* was requested. You are lucky you did not get a referencing illegal frame abort.

426 Data file already exists!

This displays when attempting to create a new DATA level on an existing dictionary using a name that already exists in the dictionary.

427 There is no data section for this file.

428 'item' object pointer not found

Results when an attempt is made to DECATALOG a PICK/BASIC program that is not compiled, or has no source item in the data section.

429 'item' program object verifies

430 'item' program object does not verify

432 'item' item-name used in program-file dictionary, compile aborted.

This means that there is an item in the DICT (dictionary) of the program file that is NOT an object code pointer item but has the same name as the source program attempting to be compiled. For example : this would occur if you tried to compile a program called "BP" in the "BP" file. If it worked, it would hammer the pointer to the DATA section.

521 Too many characters in word to block

Indicates that one of the words to block print would exceed the device width to which output is being directed.

522 BLOCK-CONVERT file missing or improperly defined

This either means that the BLOCK-CONVERT file has been deleted or that the Q-pointer from the current account is invalid.

523 Block output would exceed page width

524 The letter 'character' is missing from the block-convert file

Displayed by the BLOCK-PRINT command when a requested character is missing from the BLOCK-CONVERT file.

525 Input character 'character' is improperly formatted in BLOCK-CONVERT file

530 Already logged on

This displays when attempting to log on a line that is already on.

531 Process roadblocked

This occurs when trying to log off a line that has been marked “road-blocked” by the monitor, due to disk accesses by the process.

532 Illegal user id

This displays when trying to log a port on to an account that does not exist in the SYSTEM file.

533 Logon successful

Indicates that the LOGON verb worked.

534 Logoff successful

Indicates that the LOGOFF verb worked.

535 Illegal line number.

This displays when trying to log on a port that is outside the actual available number of ports.

536 ALREADY LOGGED OFF

This displays when logging off a line that is already logged off.

550 A required numeric parameter is missing or invalid

552 Item ‘item’ has invalid format

604 ‘label’ is an undefined label reference

630 ‘stop’ in effect. Use ‘resume’!

631 Spooler is waiting for printer cable to be attached!

632 Spooler is waiting because the printer is off-line!

633 Spooler is waiting for the tape to be detached!

634 Spooler is waiting for the printer to be detached!

637 The spooler needs the next reel of tape to continue!

638 The spooler is writing a file on tape.

640 The spooler is printing a file on the printer.

- 641** Hold file # n added;
- 644** an open file for line # n is being processed.
- 645** A closed file for line # n is being processed.
- 646** The current job has been aborted
- 651** the spooler is not active
- 652** too many pages!
- 653** No file being output or file originated on another line!
- 654** Illegal output string length!
- 655** Not attached!
- 656** End of spool queue
- 657** not a hold file
- 658** illegal number! Use sp-status
- 659** line printer ready
- 660** insufficient privileges; or device is attached to a spooler!
- 700** Run-time f-correlative abort.
- 701** Invalid function correlative definition : definition
- 705** Illegal conversion code : 'conversion'

This indicates that there is an invalid conversion code. Be careful that there are no embedded spaces in the conversion line.
- 706** the translate conversion-code : code is illegal.
- 708** 'value' cannot be converted
- 780** Item 'item' not on file.
- 781** 'item' added
- 782** 'item' updated
- 783** 'item' deleted

802 n items dumped.
 Displayed at the end of a T-DUMP or S-DUMP.

803 n item(s) loaded
 Displayed at the end of a T-LOAD.

805 n items copied

806 n items updated.

808 SYSTEM PRIORITIES
 INTERACTIVE PRIORITY priority
 BATCH PRIORITY priority
 SUPER HIGH PROCESS n
 TIME SLICE (MS) timeslice

911 Unable to run binary save—other users still logged on!

950 Linkfield error—group at n—Frame n Links: n . . .

951 Illegal forward link. group at n Frame n Links n . . .

952 Illegal backward link. group at n Frame n Links n . . .

953 Should link back to: group at n Frame n Links n . . .

954 new error items created in TSYM

990 Error in ABS-DUMP frame limits specifications

992 n items(s) have been restored

993 Account name must be specified.

999 Core PIBs Lines On PCB0 Wsstart Wssize Sysbase/mod/sep Maxfid Ovf

1004 'item' not on file

**1011 TRULY BIZARRE ERROR IN THE SELECTION PROCESSOR.
 TRY TO GENERATE A REPRODUCIBLE CASE. CALL YOUR
 SYSTEM SOFTWARE SUPPORT GROUP.**
 Yes, this is a real message.

- 1050 THAT IS AN OBSOLETE BASIC VERB (RUN OR CATALOG).
PLEASE OBTAIN THE CURRENT DEFINITION.**
- 1051 THAT IS A PROGRAM CATALOGED UNDER PRE-R80
PROTOCOLS. PLEASE RECATALOG THE PROGRAM.**
- 1059 No tape defined!**
- 1060 Illegal tape op!**
- 1065 Not detached!**
- 1080 Tape device code is illegal**
- 1081 Tape device code is undefined**
- 1082 Tape device is assigned to quarter inch Streaming Cartridge Tape**
- 1083 Tape device is assigned to low density (320K) floppy using drive 'drive'**
- 1084 Tape device is assigned to std. density (360K) floppy using drive 'drive'**
- 1085 Tape device is assigned to high density (1.2M) floppy using drive 'n'**
- 1090 ASSIGNMENT SUCCESSFUL**
- 1091 Invalid floppy drive parameter**
- 1092 Invalid floppy media type parameter**
- 1093 Requested floppy drive is not installed**
- 1094 Requested floppy drive not supported by PICK**
- 1095 Cannot set high density media in standard density drive**
- 1099 Hold entry #**
- 1100 Start code locked.**
- 1101 Null printer number.**
- 1102 Printer number too big.**
- 1103 No form number.**
- 1104 Illegal character.**

- 1105 Printer must be stopped.**
- 1106 Form number too big—exceeds 125.**
- 1107 TOO MANY PAGES IN THE PAGE SKIP—EXCEEDS 9.**
- 1108 Negative number.**
- 1109 Too many print files.**
- 1110 Illegal printer type—not P or S.**
- 1111 Illegal line number or parallel printer number.**
- 1112 Illegal parallel printer number.**
- 1113 Illegal serial printer number.**
- 1114 The device or line which you specified is being used as another printer on the system.**
- 1115 Allocation attempted on uninitialized PRINTER.**
- 1116 There is no job enqueued for output on the forms you specified. therefore, alignment is not possible.**
- 1117 Your align was just aborted by someone. Please start the align process over.**
- 1118 The printer control block has been initialized. Hopefully, the correct paper is in the printer, and the correct lpi is set.**
- 1119 You are attempting to start printer a on line n, which is not stopped.**
- 1121 You are improperly logged on.**
- 1122 Your output specification is no output. Reassign your line if you wish to output a holdfile.**

This occurs during an SP-EDIT of a hold file, choosing the “Y” response to the question, “SPOOL?”. It is because the current output assignment for the process is assigned to “hold” output. Cancel the “hold” assignment with another SP-ASSIGN, followed by NO options, and then SP-EDIT again.

- 1123 Line # n is already logged on.**

This occurs when attempting to start a serial printer on a port that is already in use by a terminal.

1126 PRINTER # n CONTROL BLOCK HAMMERED. CLEARED TO NULL.

This displays when using the SP-KILL Dn, where “n” is the number of a printer to delete from the printer queue.

1129 n Form q elements unlinked

1130 Printer list elements Stat Lk Ln Curpos Begfid Cp Fo Frms Date Time Acct

Used by LISTPEQS command.

1131 Printer list elements . # Stat Lk Ln Status Cp FoFrms Date Time Acct

Used by the LISTPEQS command.

1132 n queue elements.

1133 n frames in use.

1134 Printer assignments Printer Output queues Page Dev or Status Type Number skip line #

Used by the LISTPTR command.

1140 Your open files were closed

Displayed when using the SP-CLOSE command.

1141	Line Status	Cop	Form
	#	ies	#

1143 Align terminated; printer stopped.

1144 The tape is not available for use just now.

1145 Illegal specification number specification.

1147 Not attached.

1148 Tapeout terminated because of assign T or null assignment.

1149 Fatal spooler abort!!

1150 There is something wrong with the syntax of your verb's options.

1151 Entry # n

1160 Your output specification is no output. Reassign your line if you wish to output a holdfile.

1161 End of requested print files.

1162 End of print file control block

1169 Illegal printer number — must be between 0 and 7 inclusive.

1170 Printer # n set to stop

1171 and is inactive.

1172 but is still active.

1173 Printer # control block set to null.

1174 is unallocated.

1175 Parallel printer # n has been deleted.

1176 Serial printer # n has been deleted, and its process sent to Logon.

1177 Printer # n is inactive.

1178 The job being output on printer # n is not your print file.

1179 Job aborted on printer # n

1180 Print file # n was not unlinked because it is being output.

1181 Print file # n was not unlinked because it is unused.

1182 Print file # n was not unlinked because it is not spooled.

1183 Print file # n was not created on the account onto which you are now logged.

1184 Print file # n was unlinked and is available as a hold file.

1200 The spooler is inactive.

1201 The spooler is active.

1202 Needs to start printers

1203 Needs to log disk errors.

1204 The SPOOLER is in an unambiguous state.

See error message 1011.

1209 The control block for printer # n is in an ambiguous state. Delete the printer from the spooler system.

1210 Printer # n is

1211 unallocated.

1212 serial

1213 parallel

1214 , inactive

1215 , active

1216 , stopped

1217 , and on line.

1218 , and off line.

1219 The printer cable is off.

1220 There is no controller for this printer.

1221 The printer is defined as parallel printer # n.

1222 The printer is running on line n.

1229 Print file being output is element n

1230 Open file for line # n

1231 Closed file for line # n

1232 Generated on account

1233 , which is n frames long.

1234 and the output is choked.

1235 .

1239 Erroneously, the printer has no output queues assigned to it.

1240 Assigned output queues:


1241 ,

1242 ,

1243 The number of inter-job pages to eject is n.

1250 This system does not have parallel printers. 1

PICK/BASIC Error Messages



Some of the standard PICK/BASIC Error messages contained in this section no longer are described in the Pick documentation. They are left here for compatibility with previous versions of the system and for historical significance. Many are also good for a laugh. There are no published SMA standards for PICK/BASIC error messages.

B0 Program “*programname*” compiled. n frames used.

Displayed when a PICK/BASIC program miraculously survives the compile process.

B10 Variable has not been assigned a value; zero used!

Displayed when executing a PICK/BASIC program that references a variable that has not previously been referenced. Also occurs when writing dimensioned arrays that have not been “set” to null with a MAT assignment.

B11 Tape record truncated to tape record length!

This occurs in programs that write tape records when a tape record exceeds the number of bytes at which the tape was attached.

B12 File has not been opened

Indicates that a read or write operation was attempted on a file that has not previously been opened with an OPEN statement.

B13 Null conversion code is illegal; no conversion done!

This means that the conversion code expression in an ICONV or OCONV statement evaluated to a “null” and that it did not do exactly what was expected.

B14 Bad stack descriptor

Indicates that the number of arguments passed with a “CALL” statement differ from the number of arguments in the SUBROUTINE statement in the external subroutine. Also occurs when a file variable is used as an operand.

B15 Illegal opcode: *opcode*

Try recompiling the program.

B16 Non-numeric data when numeric required; zero used!

Typically occurs when a mathematical function is attempted on a string variable. Probably the second-most popular error message, in terms of appearance.

B17 Array subscript out-of-range

This fatal error occurs when referencing a subscript less than zero or greater than the number of subscripts (attributes) declared in the DIM or DIMENSION statement that established storage space for the dimensioned array.

B18 Attribute number less than – 1 is illegal

Occurs when the attribute expression of the READV or WRITEV statement evaluates to a negative number.

B19 Illegal pattern

Indicates a meaningless pattern in a MATCH or MATCHES statement.

B20 COL1 or COL2 used prior to executing a FIELD stmt; zero used!

This means that a reference was made to either the COL1() or COL2() functions prior to issuing a FIELD statement.

B21 Line line matread: number of attributes exceeds vector size

This occurs when the number of attributes returned in a MATREAD is greater than the number of attributes established in the DIM statement. All “extra” attributes are stuffed into the “last” defined vector, delimited by attribute marks, so at least the item will survive a MATWRITE operation. Just do not try to refer to *anything* in the last vector location. Over dimensioning is O.K.

B22 Called program is not a subroutine

Indicates that an argument of the STORAGE statement is less than 10, or not divisible by 10.

B24 Divide by zero illegal; zero used!

This indicates that a number was attempted to be divided by zero. Check the divisor to make sure that it has been assigned a value.

B25 Program programname has not been cataloged

This message displays when a CALL statement is issued, referring to an external PICK/BASIC program subroutine which has not been cataloged.

B26 UNLOCK attempted before LOCK

This indicates that an attempt was made to UNLOCK one of the 48 system execution locks prior to its having been locked.

B27 RETURN executed with no GOSUB

This typically occurs when an internal subroutine is executed without having been transferred to with a GOSUB statement, causing the RETURN statement to force this error.

B28 Not enough work space

This typically occurs when running a large PICK/BASIC program that may be dealing with one or more large data items. Program size is limited to 32,000 bytes. The solution is to break the program into smaller subroutines until this limitation is removed from the Pick system.

B29 Calling program must be cataloged

CATALOG all mainline programs and external subroutines.

B30 Array size mismatch

This occurs when a mainline program and an external subroutine both refer to the same dimensioned array, but each declares a different number of attributes. Also occurs in a “MAT copy” (MAT A = MAT B) when the number of vectors are different.

B31 Stack overflow

This occurs when a program calls too many nested subroutines.

B32 Page heading exceeds maximum of 1400 characters

A HEADING statement in PICK/BASIC cannot exceed 1400 characters.

B33 Precision declared in subprogram *programname* is different from that declared in the mainline program

This indicates that there is a PRECISION statement in an external subroutine that specifies a different number of decimal places than that of the PRECISION statement in the mainline program.

B34 File variable used where string expression expected

This indicates that some reference was made to a variable that has been declared as a file variable in an OPEN statement.

B41 Lock number is greater than 47

This means that the expression evaluated in the LOCK statement contained a number greater than 47. PICK/BASIC divides the number by 48 and the remainder is used as the lock number. Note that in release 2.2 and higher, the number of locks has been increased to 64.

B100 Compilation aborted; no object code produced

This is displayed when a compile fails for any reason. As a general rule of thumb, ignore all but the *first* message that displays when a compile fails. Find and fix the problem indicated with the first message and then recompile.

B101 Missing “END”, “NEXT”, “WHILE”, “UNTIL”, “REPEAT” OR “ELSE”; COMPILATION ABORTED, NO OBJECT CODE PRODUCED.

Not always what it appears. This can be accurate, in that there may be a missing NEXT clause in a FOR . . . NEXT sequence. Find and fix the problem indicated with the FIRST message and then recompile.

B102 Bad statement

This compile-time error indicates that there is something syntactically wrong with the line displayed immediately above this message. Look for misspelled statements and/or unclosed quotes or parentheses.

B103 Label 'label' is missing

Displayed when a GOTO statement refers to a statement label that cannot be located in the program. Make sure that the statement label is the *first* executable parameter on the line. If it follows an asterisk, for example, it will never be seen by the compiler. This also occurs when a reference to a dimensioned array is made without indicating a subscript (vector).

B104 Label 'label' is doubly defined

Indicates that there are two occurrences of the same statement label.

B105 'variable' has not been dimensioned

This displays when a non-dimensioned variable is treated as a dimensioned variable.

B106 'variable' has been dimensioned and used without subscripts

This is displayed when a reference is made to a dimensioned array without being followed by a subscript (attribute) specification.

B107 "ELSE" CLAUSE MISSING**B108 LINE line "NEXT" STATEMENT MISSING****B109 Variable missing in "NEXT" statement**

This occurs when the NEXT statement is not followed by the variable declared in the FOR statement.

B110 END statement missing

This often occurs when the END statements do not "balance" in a program, meaning that there may be a missing END statement somewhere in a series of IF-THEN clauses.

B111 "UNTIL" OR "WHILE" MISSING IN "LOOP" STATEMENT

This means that no UNTIL or WHILE conditional expression is specified in the LOOP construct.

B112 REPEAT missing in LOOP statement

This means that the REPEAT statement can not be located for the initiating LOOP statement.

B113 Terminator missing

This displays when a line containing quoted literals is missing one or more of the quote marks, or when “garbage” follows a legal statement. Also occurs at the end of a popular Arnold Schwarzenegger film.

B114 Maximum number of variables exceeded

PICK/BASIC allows for about 3,200 variables in a program. This is normally enough for most people. If not, try moving some of the variables to an external subroutine.

B115 label ‘label’ is used before the equate stmt

This occurs when a reference is made to a constant prior to its being declared with the EQU or EQUATE statement.

B116 label ‘label’ is used before the COMMON stmt

All variables must be declared in the COM or COMMON statement prior to being used in a program. This can be avoided, as can other problems, by *not* using the COMMON statement.

B117 label ‘label’ is missing a subscript list

This displays when a reference is made to a dimensioned array variable without indicating a subscript (attribute) specification.

B118 label ‘label’ is the object of an EQUATE statement and is missing

Indicates that the variable after the TO portion of an EQU statement has not been declared, or is used elsewhere in the program.

B119 Warning - precision value out of range - ignored

Indicates that a precision less than six (6) was specified, in releases 2.2 and higher. Releases 2.1 and lower supported a maximum precision of four (4).

B120 Warning - multiple precision statements - ignored!

This non-fatal error message indicates that more than one PRECISION statement is specified in the program. All but the first are ignored.

B121 Label 'label' is a constant and can not be written into.

This occurs when an attempt is made to change the value of a constant declared in an EQU or EQUATE statement.

B122 Label 'label' is improper type

Indicates an invalid expression follows the TO in an EQU or EQUATE statement.

B124 Label 'label' has literal subscripts out of range

Indicates a reference to a subscript (attribute) greater than the number of subscripts declared for the array in the DIM or DIMENSION statement; alternately, may indicate a subscript less than one.

B125 LABEL 'label' HAS A JUMP GREATER THAN 32K BYTES.

B126 OBJECT CODE EXCEEDS 65K.

You are on your own.

B127 NEXT missing

B128 LABEL 'label' EQUATED ARRAY SUBSCRIPT OUT OF RANGE.

Similar to error message B124. This means that an EQU or EQUATE statement makes a reference to a subscript (attribute) location greater than the number of subscripts defined for this array in its DIM or DIMENSION statement.

B130 LINE line THERE IS AN UNRATIONALIZABLE EQUATE.

B154 FOR STATEMENT WITH NO NEXT STATEMENT

Each FOR . . statement *must* be terminated with a NEXT . . If it is, then look at the statements between the FOR and the NEXT to see if there is an extra END statement that would logically cause the compile to think that the end of the program had been reached.

B209 File is update protected

Indicates that an update (write operation) was attempted on an update-restricted file.

B210 File is access protected

Indicates that a read operation was attempted on a read-restricted file.

B222 'CSYM' is not a file name or needs a data level

This displays when the pointer to the CSYM file is missing or improperly defined in the MD of the current account.

ASCII Table



<i>ASCII Character</i>	<i>Decimal</i>	<i>Hex</i>	<i>EBCDIC</i>	<i>Keyboard Entry</i>
NUL	0	00	00	
SOH	1	01	01	<ctl>A
STX	2	02	02	<ctl>B
ETX	3	03	03	<ctl>C
EOT	4	04	37	<ctl>D
ENQ	5	05	2D	<ctl>E
ACK	6	06	2E	<ctl>F
BEL	7	07	2F	<ctl>G
BS	8	08	16	<ctl>H
HT	9	09	05	<ctl>I
LF	10	0A	25	<ctl>J
VT	11	0B	0B	<ctl>K
FF	12	0C	0C	<ctl>L
CR	13	0D	0D	<ctl>M
S0	14	0E	0E	<ctl>N
SI	15	0F	0F	<ctl>O
DLE	16	10	10	<ctl>P
DC1	17	11	11	<ctl>Q (X-ON)
DC2	18	12	12	<ctl>R

<i>ASCII Character</i>	<i>Decimal</i>	<i>Hex</i>	<i>EBCDIC</i>	<i>Keyboard</i>
DC3	19	13	3A	<ctl>S (X-OFF)
DC4	20	14	3C	<ctl>T
NAK	21	15	3D	<ctl>U
SYN	22	16	32	<ctl>V
ETB	23	17	26	<ctl>W
CAN	24	18	18	<ctl>X
EM	25	19	19	<ctl>Y
SUB	26	1A	3F	<ctl>Z
ESC	27	1B	27	ESCAPE KEY
FS	28	1C	1C	
GS	29	1D	1D	
RS	30	1E	1E	
US	31	1F	1F	
(blank)	32	20	40	
!	33	21	5A	
“	34	22	7F	
#	35	23	7B	
\$	36	24	5B	
%	37	25	6C	
&	38	26	50	
\	39	27	7D	
(40	28	4D	
)	41	29	5D	
*	42	2A	5C	
+	43	2B	4E	
,	44	2C	6B	
-	45	2D	60	
.	46	2E	4B	
/	47	2F	61	
0	48	30	F0	
1	49	31	F1	
2	50	32	F2	
3	51	33	F3	
4	52	34	F4	
5	53	35	F5	
6	54	36	F6	
7	55	37	F7	
8	56	38	F8	
9	57	39	F9	
:	58	3A	7A	
;	59	3B	5E	
<	60	3C	4C	
=	61	3D	7E	

<i>ASCII Character</i>	<i>Decimal</i>	<i>Hex</i>	<i>EBCDIC</i>	<i>Keyboard Entry</i>
>	62	3E	6E	
?	63	3F	6F	
@	64	40	7C	
A	65	41	C1	
B	66	42	C2	
C	67	43	C3	
D	68	44	C4	
E	69	45	C5	
F	70	46	C6	
G	71	47	C7	
H	72	48	C8	
I	73	49	C9	
J	74	4A	D1	
K	75	4B	D2	
L	76	4C	D3	
M	77	4D	D4	
N	78	4E	D5	
O	79	4F	D6	
P	80	50	D7	
Q	81	51	D8	
R	82	52	D9	
S	83	53	E2	
T	84	54	E3	
U	85	55	E4	
V	86	56	E5	
W	87	57	E6	
X	88	58	E7	
Y	89	59	E8	
Z	90	5A	E9	
[91	5B	80	
\	92	5C	E0	
]	93	5D	90	
^	94	5E	5F	
_	95	5F	6D	
`	96	60	79	
a	97	61	81	
b	98	62	82	
c	99	63	83	
d	100	64	84	
e	101	65	85	
f	102	66	86	
g	103	67	87	
h	104	68	88	

<i>ASCII Character</i>	<i>Decimal</i>	<i>Hex</i>	<i>EBCDIC</i>	<i>Keyboard Entry</i>
i	105	69	89	
j	106	6A	91	
k	107	6B	92	
l	108	6C	93	
m	109	6D	94	
n	110	6E	95	
o	111	6F	96	
p	112	70	97	
q	113	71	98	
r	114	72	99	
s	115	73	A2	
t	116	74	A3	
u	117	75	A4	
v	118	76	A5	
w	119	77	A6	
x	120	78	A7	
y	121	79	A8	
z	122	7A	A9	
{	123	7B	C0	
	124	7C	6A	
}	125	7D	D0	
~	126	7E	A1	
DEL	127	7F	07	

The following are high-order ASCII characters:

128	80	04
129	81	06
130	82	08
131	83	09
132	84	0A
133	85	13
134	86	14
135	87	15
136	88	17
137	89	1A
138	8A	1B
139	8B	20
140	8C	21
141	8D	22
142	8E	23
143	8F	24
144	90	28

<i>ASCII Character</i>	<i>Decimal</i>	<i>Hex</i>	<i>EBCDIC</i>	<i>Keyboard Entry</i>
	145	91	29	
	146	92	2A	
	147	93	2B	
	148	94	2C	
	149	95	30	
	150	96	31	
	151	97	33	
	152	98	34	
	153	99	35	
	154	9A	36	
	155	9B	38	
	156	9C	39	
	157	9D	3B	
	158	9E	3E	
	159	9F	41	
	160	A0	42	
	161	A1	43	
	162	A2	44	
	163	A3	45	
	164	A4	46	
	165	A5	47	
	166	A6	48	
	167	A7	49	
	168	A8	4A	
	169	A9	4F	
	170	AA	51	
	171	AB	52	
	172	AC	53	
	173	AD	54	
	174	AE	55	
	175	AF	56	
	176	B0	57	
	177	B1	58	
	178	B2	59	
	179	B3	62	
	180	B4	63	
	181	B5	64	
	182	B6	65	
	183	B7	66	
	184	B8	67	
	185	B9	68	
	186	BA	69	
	187	BB	70	

<i>ASCII Character</i>	<i>Decimal</i>	<i>Hex</i>	<i>EBCDIC</i>	<i>Keyboard Entry</i>
	188	BC	71	
	189	BD	72	
	190	BE	73	
	191	BF	74	
@	192	C0	75	
A	193	C1	76	
B	194	C2	77	
C	195	C3	78	
D	196	C4	8A	
E	197	C5	8B	
F	198	C6	8C	
G	199	C7	8D	
H	200	C8	8E	
I	201	C9	8F	
J	202	CA	9A	
K	203	CB	9B	
L	204	CC	9C	
M	205	CD	9D	
N	206	CE	9E	
O	207	CF	9F	
P	208	D0	A0	
Q	209	D1	AA	
R	210	D2	AB	
S	211	D3	AC	
T	212	D4	AD	
U	213	D5	AE	
V	214	D6	AF	
W	215	D7	B0	
X	216	D8	B1	
Y	217	D9	B2	
Z	218	DA	B3	
[219	DB	B4	
\	220	DC	B5	
]	221	DD	B6	
^	222	DE	B7	
_	223	DF	B8	
`	224	E0	B9	
a	225	E1	BA	
b	226	E2	BB	
c	227	E3	BC	
d	228	E4	BD	
e	229	E5	BE	
f	230	E6	BF	

<i>ASCII Character</i>	<i>Decimal</i>	<i>Hex</i>	<i>EBCDIC</i>	<i>Keyboard Entry</i>
g	231	E7	CA	
h	232	E8	CB	
i	233	E9	CC	
j	234	EA	CD	
k	235	EB	CE	
l	236	EC	CF	
m	237	ED	DA	
n	238	EE	DB	
o	239	EF	DC	
p	240	F0	DD	
q	241	F1	DE	
r	242	F2	DF	
s	243	F3	E1	
t	244	F4	EA	
u	245	F5	EB	
v	246	F6	EC	
w	247	F7	ED	
x	248	F8	EE	
y	249	F9	EF	
z	250	FA	FA	
[251	FB	FB	START BUFFER
\	252	FC	FC	SUBVALUE MARK
]	253	FD	FD	VALUE MARK
^	254	FE	FE	ATTRIBUTE MARK
—	255	FF	FF	SEGMENT MARK

CompuSheet⁺



An important feature of CompuSheet⁺ is its extensive on-line help system. A detailed explanation of any prompt can be obtained by entering a question mark followed by a <cr> at the prompt. This context-sensitive help system will, in many cases, eliminate the need for a user manual for most functions of CompuSheet⁺.

ENTERING COMPUSHEET⁺

The CompuSheet⁺ spreadsheet is evoked from the CS-MENU routine or from entering COMPUSHEET from TCL. Upon entering the program, the following prompts appear:

File name:

Enter the name of the file containing the spreadsheet to be processed or the file which will contain a newly created spreadsheet. This file should have been created via the CS-MENU routine and should be designated as reserved for spreadsheets.

➡ Spreadsheet Name

Enter the name of the spreadsheet to be processed. This must be the name of either an existing spreadsheet in this file or the name of a spreadsheet to be

created. The name must not contain any spaces and must not be longer than 15 characters.

➡ Password

If this is a new spreadsheet, it can be optionally encoded with a password. Be sure to make a note of any password since the password will be required in order to recall the spreadsheet. If this is an existing spreadsheet that has been password encoded, a valid password must be entered here. If this is an existing spreadsheet that has not been password encoded, this prompt will be by-passed.

If this is an existing spreadsheet, the description will be displayed and the following prompts will appear:

Do you want to change the description? (Y or N=<cr>):

Enter a Y to change the description via the CompuSheet⁺ editor. Enter an N or <cr> to by-pass changing the description.

Last update: By: nnn On: dd mmm yyyy At:
hh:mm:ss

Information is displayed pertaining to the last update, including the name or initials of the operator, the date of the last update, and the time of the last update.

Please enter your initials/name or "/X":

The initials or name entered here will be recorded as the individual who last updated the spreadsheet. Enter /X to start over at the Spreadsheet name: prompt. If this is a new spreadsheet, the following prompts will appear:

Description:

Enter up to 10 lines of 60 characters each to describe this spreadsheet. This description is for documentation purposes. At least one character of description must be entered. It is suggested that this information include the purpose of the spreadsheet, a date on which the spreadsheet can be removed from the system, and any other data that will help in maintaining the spreadsheet. The following prompt will appear:

Please enter your initials/name or "/X":

The initials or name entered here will be recorded as the individual who last updated the spreadsheet. Enter /X to start over at the Spreadsheet name: prompt.

New spreadsheet — file name: filename spread
sheet name: spreadsheetname

Enter Y if you wish to define you own spreadsheet format or <cr> to use standard spreadsheet format:

Enter a Y to define the layout and format of the new spreadsheet as described below. If a <cr> is entered, a standard layout of 14 columns will be displayed. If a selection to define the spreadsheet format was made, the following prompts will appear:

```
. . . Define spreadsheet size and format . . .  
From COL    Thru COL    WIDTH    FORMAT  
1
```

Enter the column range being defined. "From COL" is filled in by CompuSheet⁺, enter the "Thru COL" with the ending column of the range being defined, "WIDTH" with the width of each column in the range, and "FORMAT" with the format mask to be assigned to each column in the range (refer to the FORM command for an explanation of valid column formats or enter a question mark (?) for examples). These prompts are repeated until all columns are defined.

After all columns have been defined, the entry of a <cr> at the "Thru COL" prompt will cause the blank spreadsheet to be displayed.

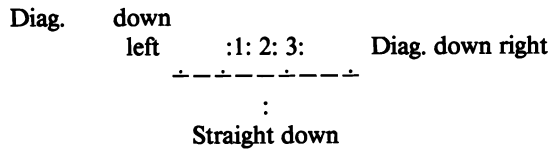
COMMAND MENU

CompuSheet⁺ contains a command menu system that can be called by depressing the space bar at the "Command:" prompt. The first level of the command menu displays a list of command categories which are highlighted by subsequent depressions of the space bar. Entry of <cr> at a highlighted category causes display of the associated command list, with the various commands being highlighted by subsequent depressions of the space bar. To select a command, enter <cr> at a highlighted command.

CURSOR CONTROL

The cursor is controlled by the 10-key keypad as explained below:

```
                Straight up  
                :  
                - - - - -  
Diag. up left  :7: 8: 9:  Diag. up right  
                ÷ - - - - ÷  
left          :4: 5: 6:  right  
                ÷ - - - - ÷
```



5 key = "jump" to a specified cell in this spreadsheet or into another spreadsheet
 0 key = "page" in direction indicated by the following cursor key
 (1,2,3,4,6,7,8,9)

ENTERING DATA, FORMULAS, AND HEADINGS

Once the cursor is "pointing" to the cell where data, formulas, or a heading is to be entered, one of the following commands provide for the entry:

- . (period) Enters the mode for entry into a single cell and then returns to the Command: prompt.
- , (comma) Enters the mode for entry into a range of cells according to the setting of the input direction indicator (see IDir command). The entry mode is maintained until no data or formula is entered into a cell.

Upon depression of either of the commands, the cursor moves to the highlighted cell for entry of data. If no data is entered (the depression of <cr>), a prompt for a formula will occur. If no formula is entered (the depression of a <cr>), the cursor returns to the Command: prompt. The entry of a colon (:) as the first character of data (or the first character of the result of a formula) will display the associated data as a heading (across cell boundaries). After entry into the cell, the cursor will move to the next cell according to the setting of the input direction indicator (see the IDIR command). The entry of a backslash (\) as the first character of data will cause the contents of the cell (data and formula) to be cleared.

EDITOR

CompuSheet⁺ utilizes a common line editor routine for editing all spreadsheet descriptions, macros, rules, formulas, data, and headings. A common set of commands control the editor:

➤ Editor Cursor Movement Commands

- 1 moves the cursor to the start of the edited text
- 3 moves the cursor to the end of the edited text

- 4 moves the cursor one character to the left
- 6 moves the cursor one character to the right
- 7 moves the cursor ten characters to the left
- 9 moves the cursor ten characters to the right

➤ Editor Commands

C

COPY CELL CONTENTS. Copies the contents of another cell (data, heading, or formula) into the edited cell. Do not use this command to copy cells into spreadsheet descriptions, rules, or macros.

D

DELETE CHARACTER AT CURSOR. Deletes the character at the cursor.

\

DELETE THE ENTIRE STRING OF TEXT. The entry of a backslash character at any point in a string causes the entire string to be deleted.

E

EXIT THE EDITED STRING. Exits the editor without applying any changes made this editing session.

I

INSERT TEXT AT CURSOR. Enters a mode providing for the insertion of text in front of the cursor. Any existing text following the insertion point is shifted to the right. Entering a <cr> exits the insertion mode.

O

OVERTYPE TEXT AT CURSOR. Enters a mode providing for overtyping existing text starting at the cursor. Entering a <cr> exits the overtype mode.

S

(or <cr>). **SAVE ALL CHANGES AND EXIT THE EDITOR.** Saves all changes made during the editing session and exits the editor.

R

REPLACE. Replaces one string of characters with another string of characters. The following prompts will appear:

Enter string to replace: Enter the character string to be replaced

Replace with: Enter the replacement character string or <cr> to delete the replaced string.

Occurrence to change or <cr>=1, ""*=all: Enter the occurrence to replace. A <cr> or 1 is treated as the first occurrence, 2 replaces the second occurrence, etc. An asterisk replaces all occurrences.

X

CANCEL ALL EDIT CHANGES THIS SESSION. Cancels all changes made this editing session and restores the text to the original form.

Z

DELETES FROM THE CURSOR TO END OF TEXT. Deletes all characters starting at the cursor thru the end of the line of text.

COMPUSHEET⁺ COMMANDS

The commands that control CompuSheet⁺ can be evoked by entering the first two characters of the command, by entering the entire command, or via the command menu described above. The following is an explanation of each of the commands that control CompuSheet⁺. For details of the operation of any of the commands, refer to the user manual or utilize the on-line help by entering a question mark (?).

=

(equal). Initiates the execution of a MACro. Upon entry of the command, a prompt is made for the macro name. Upon entry of the name the macro will begin execution.

AU

Audit. Prints a comprehensive spreadsheet audit report on either the line printer or a terminal driven slave printer. The audit report is formatted and can contain all data (in internal and external formats), formulas, formatting information, macro names, rules, initials, and date of last update, etc.

CA

Calc. Causes all formulas in the spreadsheet to be calculated according to the calculation direction indicator (see CDir) and the spreadsheet is redisplayed.

CD

(CDir). Toggles the calculation direction indicator (the second entry following “Dir:” on status line 1) between “R” and “C”. A setting of “R” causes formulas to be calculated across each row, starting with row 1 and ending with the last row on the spreadsheet. A setting of “C” causes formulas to be calculated from top to bottom of each column, starting with column 1 and ending with the last column on the spreadsheet.

CL

Clear. Clears data, formulas, or both from any range(s) of cells. Prompts are made for the range of cells (upper left and lower right) to be cleared and whether data, formulas, or both are to be cleared. Multiple ranges of cells can be specified with one command.

CO

Copy. The copy command allows for the copying of data, formulas, or both from one section of a spreadsheet to another. The command provides for copying:

- ☐ one cell to another cell
- ☐ one cell to a range of other cells (a step value may be specified providing for copying a cell to every other, third, etc. cell in the range)
- ☐ a column or part of a column to another column or range of columns (with step value an option)
- ☐ a row or part of a row to another row or range of rows (with step value an option)
- ☐ a block of cells to another part of a spreadsheet

DE

Delete. Deletes a set of one or more columns or rows from a spreadsheet. Any remaining columns or rows will be “pulled in” to fill in the gap. Optionally, formulas affected by the deletion may be adjusted.

DI

Display. Causes the screen image of the spreadsheet to be redisplayed. No change is made to the spreadsheet.

ED

Edit. Provides for editing the contents of any cell. The data in a cell can be edited (unless it is the result of a formula) or any formula can be edited. Refer to the “Editor” section above for the commands that control the editor.

EX

Exit. Causes the spreadsheet session to be terminated **WITHOUT** saving the changes made this session. A prompt requesting verification is made prior to exiting the spreadsheet.

FB

(Fbld). Writes data from a spreadsheet to a standard file. Options provide for the output of columns (or parts of columns) to file items, or for the output of rows (or parts of rows) to file items where each column or row is converted into one file item. Options provide for building file dictionary items describing the data and for either the specification or the generation of item-ids. For security purposes, the receiving file must be authorized for the use of FBld.

FI

File. Writes the spreadsheet to the file or deletes the spreadsheet. Options include:

- ☐ Write the spreadsheet to the current file under the current name and password (if any)
- ☐ Write the spreadsheet to the file while adding or changing or deleting a password
- ☐ Write the spreadsheet to the current file or another file under either the current name and password or a new name and password. The original spreadsheet is not changed
- ☐ Delete the spreadsheet from the file

FL

(Fload). Provides a means to load the contents of any file directly into the spreadsheet. A request is made for the name of the file to be loaded, the row of the spreadsheet where the loaded data is to begin, the file dictionary item names to be loaded and the column numbers where they are to be loaded, and any selection and sorting criteria. FL uses the ACCESS processor to pass the data to the spreadsheet and therefore follows ACCESS conventions for selection and sorting criteria, and conversions and correlatives.

FO

Form. Controls the display and printing format by columns or by individual cells. The format consists of the justification (alignment to the left or right cell margins) and the way numeric data appears (decimal points, commas, credit representations . . .).

GR

Graph. Prints graphs of data on the spreadsheet. Data in columns or rows can be graphed as bar charts (including stacked or clustered bars), line charts,

scatter (point-to-point) diagrams, and pie charts. ACCU/PLOT must be installed in order to use the GR command.

ID

(Idir). Toggles the setting of the input direction indicator (the first value following "Dir:" on the status line). During data/formula entry, the window cursor moves from cell to cell according to this value. If set to "R", the next cell selected will be on the same row, to the right of the current cell. If set to "C", the next cell selected will be in the same column, below the current cell.

IF

(Iform). Toggles the "Data:" display on status line 2. This data is displayed in the internal format (unrounded up to 4 decimal places, and without formatting characters).

IN

Insert. Provides for inserting or adding new columns or rows to a spreadsheet. One or more columns or rows can be inserted with one command. Optionally, formulas affected by the insertion can be adjusted.

LO

Lock. Provides the means to lock or protect a cell or range of cells from data/formula entry. The command provides for setting and removing the lock on cells. Note: Lock does not protect cells from commands like COpY, CLear, MErge, etc.

MA

Macro. Records and names a series of keystrokes which can be "played back" by name. This command starts and stops the recording process. It also can be used to execute, edit, overwrite, or delete a macro.

ME

Merge. Provides for retrieving a block of cells from another spreadsheet and placing the block into the current spreadsheet. This block can be copied into the receiving spreadsheet, overlaying any existing data, or it can be added or subtracted from any existing numeric values in the receiving spreadsheet. Data, formulas, or both may be merged.

NE

Next. Controls the window cursor during the "comma" entry mode. The next cell selected for entry is normally determined by the setting of the input

direction indicator (see IDir). The NExt command is used to override the input direction indicator by specifying the linking of cells to be selected for entry.

PA

Page. Toggles the setting of the “page” indicator (the third value following “Dir:” on the status line). If set to a “P”, when a window margin is “bumped” by the cursor, the next page (group of columns or rows) will be displayed. If set to a “N”, when a window margin is “bumped” by the cursor only the next column or row is displayed.

PR

Print. Prints one or more copies of the spreadsheet on the line printer or a terminal driven slave printer. Options provide for printing the entire spreadsheet, letting CompuSheet+ calculate the points where page breaks occur; or for user selection of blocks of data to be printed and the point where page breaks occur. Specific columns or rows containing heading information can be repeated on each page.

RC

(RCalc). Provides for calculating a specified range of cells independently of others. The user can specify the block to be calculated and the direction of the calculation. Multiple blocks may be specified at one time.

RS

(RSort). Allows sorting any block of cells within a spreadsheet by either column or row. The upper left and lower right cells in the block are specified along with the column(s) or row(s) to be used as the sort keys. Keys can be sorted in either ascending or descending sequence.

RU

Rule. Equates an expression (formula) to a name. Names can be equated to a cell, a column, a row, or any valid expression. The assigned names can be used in any formula or other rule in place of the associated expression. At execution time, the name is replaced with the expression.

SA

Save. Saves the current spreadsheet to the file without exiting. Options include:

- ☐ Saving the spreadsheet to the current file under the current name and password (if any)

- ☐ Saving the spreadsheet to the file while adding or changing or deleting a password
- ☐ Saving the spreadsheet to the current file or another file under either the current name and password or a new name and password. The original spreadsheet is not changed

SO

Sort. Sorts a specified set of rows on the spreadsheet into a sequence determined by the contents of one or more columns. The key columns can be sorted in either ascending or descending sequence.

SR

Search. Searches the spreadsheet for a specified set of characters. The entire spreadsheet can be searched, or a specific range of cells can be searched. The search will pause at each cell containing a matching set of characters and if requested, will continue searching for the next match.

ST

Stats. Displays the spreadsheet status screen containing information regarding the current file and spreadsheet names, information on last update, spreadsheet size, current window settings, current terminal definition, etc.

TE

Term. Calls the terminal definition screen for the current port number. This command is used to define terminal and printer characteristics for the port number in use. The type of terminal in use, screen width and depth, printer width and depth, and visual attributes (such as reverse video, color, etc.), is specified with this process. Also, users can equate CompuSheet+ commands with keys on the keyboard such as function keys, cursor control keys, etc.

WI

Width. Provides the means to change the width of any column(s) on the spreadsheet. Options provide for changing the width of a single column or a range of columns (with provision of a step value).

WN

Window. Allows for splitting the screen into vertical (by column) and horizontal (by row) windows. With the WN command, the screen can display different parts of the spreadsheet at the same time. Up to 10 sets of columns and 10 sets of rows can be displayed at one time. As changes are

made in one area of the spreadsheet, the effect upon other parts can be viewed.

XE

(Xeq). Initiates a temporary exit from the spreadsheet to a pseudo TCL prompt. From this prompt, users can run any TCL process as long as they do not LOGTO another account or log OFF.

FORMULA ENTRY

Formulas are entered at the Formula: prompt. This prompt is displayed as part of the data/formula entry sequence and is initiated by the period (.) and comma (,) commands. Upon entry of either of these commands, a prompt is made for data in the window cursor. If no data is entered and <cr> is depressed, a prompt is made for a formula on status line 2. If a formula had previously been entered into the cell, the period or comma commands will immediately transfer to the formula editor without prompting for data.

ERRORS IN FORMULAS

There are two types of errors that occur in formulas. The first involves “syntax errors” or errors in the formula structure and include missing quotes, missing commas, missing parentheses, invalid operators, misspelled functions, etc. When these types of errors occur, the formulas editor will detect the error and will, as closely as possible, inform the user of the error in structure. The second type of error involves formulas that are structurally correct but attempt invalid operations such as division by zero, arithmetic operation on alphabetic data, and other similar invalid operations that are detected at the time of calculation. These types of errors return an error message consisting of “ERRxxx” where “xxx” is an error number. Error numbers are explained in the user manual, in the help screens, and if the user places the window cursor on the cell in error and displays the internal format (IForm), an abbreviated error message is displayed in the “Data:” display.

PRECEDENCE AND PRECISION

The precedence of operators follow the rules of algebra. Infinite levels of parentheses are allowed to specify the order of operations. In the absence of parentheses, multiplication and division have precedence over addition and subtraction, which in turn have precedence over relational operators.

Decimal precision is carried out to four (4) decimal places except for the financial calculation functions (FC, IRR, NPV) which use 12 decimal place precision.

REFERENCING SPREADSHEET CELLS, COLUMNS AND ROWS

Cells are referenced by their intersecting column and row numbers. The format of a cell reference is col.row where col is any column number and row is any row number. A period separates the column and row numbers. The column number or row number can be omitted for any reference (such as .row or col.) and the column or row occupied by the formula will be substituted at the time of calculation. Valid cell references might be 2.5 which references the cell where column 2 intersects with row 5. Other examples are .5 which references the cell where the column containing the formula intersects with row 5; and 2. which references the cell where column 2 intersects with the row containing the formula.

Either the column or row reference may be in the format Pn or Fn where the “P” indicates previous and “F” indicates following, and “n” indicates the number of previous or following columns or rows. Examples are P3.5 (the cell three columns prior to the column containing the formula and on row 5), 5.P3 (the cell in column 5 three rows up from the row containing the formula), P1. (the cell in the previous column on the row containing the formula), P1.F5 (the cell in the previous column and five rows below the cell containing the formula).

LITERALS

A *literal* is any numeric or series of alphabetic characters that do not reference a cell, column number, or row number. It indicates “use this specific value” in contrast to a reference to a cell containing a value to be used. All literals must be enclosed in either single or double quote signs. A literal can be a number such as “1234.56” or “-100” or “.005” or a series (string) of characters such as “ABCD” or “12b3” or “OVER LIMIT” or “*****”.

BASIC ARITHMETIC OPERATORS

The basic arithmetic operators are: + (for addition), - (for subtraction), * (for multiplication) and / (for division). Examples are 2.5 + 3.5 or 2. - 3. or .P2*“1.3333” or .12/(.10+.11)

FUNCTIONS

Functions provide “short cuts” to performing common types of calculations. These include:

SUM(*cell1*,*cell2*{,*exp1*{,*exp2*}})

Summation (returns the sum of a range of cells) The meanings of the variables are:

- cell1* the starting cell number in the range (upper left)
- cell2* the ending cell number in the range (lower right)
- exp1* optional step value (2=sum every other cell, 3=sum every third cell, etc.)
- exp2* optional indicator where “+”=sum only positive values, “-”=sum only negative values

AVG(*cell1*,*cell2*{,*exp1*{,*exp2*}})

Average (returns the average of a range of cells) The meanings of the variables are:

- cell1* the starting cell number in the range (upper left)
- cell2* the ending cell number in the range (lower right)
- exp1* optional step value (2=every other, 3=every third, etc.)
- exp2* optional indicator where “+”=average only positive values, “-”=average only negative values, “S”=do not include zero values in the average

CNT(*cell1*,*cell2*{,*exp1*{,*exp2*}})

Count (returns a count of the numeric values in a range of cells) The meanings of the variables are:

- cell1* the starting cell number in the range (upper left)
- cell2* the ending cell number in the range (lower right)
- exp1* optional step value (2=every other, 3=every third, etc.)
- exp2* optional indicator where “+”=count only positive values, “-”=count only negative values, “S”=do not include zero values in the count

MIN(*cell1*,*cell2*{,*exp1*{,*exp2*}})

Minimum (returns the minimum value in a range of cells) The meanings of the variables are:

- cell1* the starting cell number in the range (upper left)
- cell2* the ending cell number in the range (lower right)
- exp1* optional step value (2=every other, 3=every third, etc.)
- exp2* optional indicator where “+”=use only positive values, “-”=use only negative values, “S”=do not include zero values

MAX(*cell1*,*cell2*{,*exp1*{,*exp2*}}))

Maximum (returns the maximum value in a range of cells) The meanings of the variables are:

cell1 the starting cell number in the range (upper left)
cell2 the ending cell number in the range (lower right)
exp1 optional step value (2 = every other, 3 = every third, etc.)
exp2 optional indicator where “+” = use only positive values, “-” = use only negative values, “S” = do not include zero values

IF(*expC*,*expT*,*expF*)

Conditional (test a condition and, if true execute one expression, if false execute another) The meanings of the variables are:

expC the conditional expression — operators are: = or EQ (equal), # or NE or <> (not equal), < or LT (less than), > or GT (greater than), <= or LE (less than or equal), >= or GE (greater than or equal), AND, OR, NOT(*exp*), NUM(*exp*), ALPHA(*exp*)
expT the expression to execute if the conditional expression evaluates to “true”
expF the expression to execute if the conditional expression evaluates to “false”

MATH AND TRIG (miscellaneous math and trig functions)

SIN(<i>exp</i>)	returns the sine of <i>exp</i> in degrees
COS(<i>exp</i>)	returns the cosine of <i>exp</i> in degrees
TAN(<i>exp</i>)	returns the tangent of <i>exp</i> in degrees
LN(<i>exp</i>)	returns the natural (base e) logarithm of <i>exp</i>
EXP(<i>exp</i>)	raises the number ‘e’ (2.1783) to the value of <i>exp</i>
SQRT(<i>exp</i>)	returns the square root of <i>exp</i>
PWR(<i>exp1</i> , <i>exp2</i>)	raises <i>exp1</i> to the power of <i>exp2</i> (if <i>exp2</i> =0 then 1 is returned)
REM(<i>exp1</i> , <i>exp2</i>)	returns the remainder of <i>exp1</i> divided by <i>exp2</i>
ABS(<i>exp</i>)	returns the absolute value of <i>exp</i>
INT(<i>exp</i>)	returns the integer portion of <i>exp</i>
RD(<i>exp1</i> , <i>exp2</i>)	returns the result of <i>exp1</i> rounded to <i>exp2</i> places (limit of 4)
RND(<i>exp</i>)	returns a random number between zero and one less than <i>exp</i>

STRING OPERATIONS (functions that manipulate strings)

exp1:exp2

Concatenates (connects) exp1 to exp2

SUB(*exp1,exp2,exp3*)

Returns a substring of exp1 starting at exp2 for a length of exp3

exp1[exp2,exp3]

Returns a substring of exp1 starting at exp2 for a length of exp3

LEN(*exp*)

Returns the length of exp in characters

FIELD(*exp1,exp2,exp3*)

Returns a substring of exp1 using exp2 as the delimiter and exp3 as the occurrence to return

INDEX(*exp1,exp2,exp3*)

Returns the beginning location of exp3 occurrence of exp2 in exp1

SPACE(*exp*)

Returns a string of spaces exp long

STR(*exp1,exp2*)

Returns a string of exp1, repeated exp2 times

TRIM(*exp*)

Removes all extraneous spaces from exp

READING INFORMATION FROM THE DATABASE

READ(*"filename",exp1,exp2{,exp3{,exp4}}*)

Returns an attribute, value, or subvalue from an item in a file, where: *filename* is the name of the file to be read and must be a literal enclosed in quotes (*filename* or *dictname,datafilename*)

- exp1* is the item-id and can be the number of a cell containing the item-id, a literal of the item-id, or any expression resulting in an item-id. If *exp1* is N (no quotes) the item-id will be obtained from the next available item-id on a select list
- exp2* an expression that returns the attribute number in the item to be returned (a cell number, literal, or expression resulting in the attribute number)
- exp3* an expression that returns the value number in the attribute to be returned (a cell number, literal, or expression resulting in the value number) and is optional
- exp4* an expression that returns the subvalue number in the value to be returned (a cell number, literal, or expression resulting in the subvalue number) and is optional

GET(*exp1*{*exp2*(*exp3*)})

Returns an attribute, value, or subvalue from the last item retrieved by a READ function where:

- exp1* an expression that returns the attribute number in the item to be returned (a cell number, literal, or expression resulting in the attribute number)
- exp2* an expression that returns the value number in the attribute to be returned (a cell number, literal, or expression resulting in the value number) and is optional
- exp3* an expression that returns the subvalue number in the value to be returned (a cell number, literal, or expression resulting in the subvalue number) and is optional

RETRIEVING DATA FROM OTHER SPREADSHEET CELLS

READW("filename",*exp1*,*exp2*)

Reads the contents of a cell from another spreadsheet where: *filename* is the name of the file containing the spreadsheet to be read and must be a literal enclosed in quotes

- exp1* an expression resulting in the name of the spreadsheet to be read
- exp2* an expression resulting in the number of the cell to be returned

GETW(*exp*)

Returns a cell from the spreadsheet last retrieved by a READW function

- exp* an expression resulting in the number of the cell to be returned

TABLE LOOKUP

FIND(*cell1,cell2,exp1,exp2*)

Performs a lookup in a table contained within the current spreadsheet where:

- cell1* the first cell of the range of cells to search and must be cell number (col.row) rather than an expression
- cell2* the last cell of the range of cells to search and must be a cell number (col.row) rather than an expression
- exp1* the value to be located in the table and can be any expression: a literal, a cell number, or a formula
- exp2* the offset number of columns or rows containing the value to return (the number of columns or rows AWAY from the range where the value to be returned is located — can be negative)

FINDW(*"filename",exp1,exp2,exp3,exp4,exp5*)

Performs a table lookup in a table contained in a spreadsheet other than the current spreadsheet where: *filename* is the name of the file where the spreadsheet containing the table is stored (must be in the form of a literal within quotes and cannot be in the form of a cell number or expression)

- exp1* the name of the spreadsheet containing the table to be used by the function and can be any expression
- exp2* the first cell of the range of cells in the table to be searched and can be any expression
- exp3* the last cell of the range of cells in the table to be searched and can be any expression
- exp4* the value to be located in the table and can be any expression
- exp5* the offset number of columns or rows containing the value to return (the number of columns or rows AWAY from the range where the value to be returned is located — can be negative)

MISCELLANEOUS

- T** returns the system time in internal format
- D** returns the system date in internal format
- N** returns the next item-id off of a select list

ICONV(*exp1,exp2*)

Calculates *exp1* and performs an input conversion as defined by *exp2*

OCONV(*exp1,exp2*)

Calculates *exp1* and performs an output conversion as defined by *exp2*

COLOR(*exp1,exp2*)

Calculates *exp1* and displays the result in color as defined by *exp2* which can be any number from 31 thru 97 as defined for the terminal in use (refer to the terminal definition routine)

CALLING PICK/BASIC SUBROUTINES FROM FORMULAS

CALL(*exp1,exp2{,exp3 . . . ,exp9}*)

Calls a basic subroutine from a formula where:

exp1 identifies the subroutine to be called (it must be cataloged)
exp2 any valid expression and is passed as argument 1 of the subroutine
exp3 thru *exp9* are optional and, if used, must be valid expressions and are passed to the subroutine as arguments 2 thru 8

FINANCIAL

FC(*exp1,exp2,exp3,exp4,exp5,exp6,exp7*)

Financial calculation function which solves for an unknown when other values are known (enter a question mark in quotes in place of the expression to be solved) where:

exp1 annuity type (S): 0 = ordinary annuity 1 = annuity due
exp2 number of periods or years (N)
exp3 interest rate per period or year in decimal form (R)
exp4 future value (F)
exp5 present value (P)
exp6 annuity payment in equal payments per period (A)
exp7 compounding factor: 1 = annual, 4 = quarterly, 12 = monthly, 360 or 365 = daily (C)

NPV(*cell1,cell2,exp1,exp2*)

Calculates the net present value of a stream of cash flows where:

cell1 the first cell containing the stream of cash flows
cell2 the ending cell containing the stream of cash flows

- exp1* the initial investment amount and must be greater than zero
- exp2* the interest rate or assumed rate of return and must be in the range from zero (0) to one (1)

IRR(*cell1,cell2,exp1*)

Calculates the internal rate of return of a stream of cash flows where:

- cell1* the first cell containing the stream of cash flows
- cell2* the ending cell containing the stream of cash flows
- exp1* the investment amount

REFERENCING OTHER FORMULAS

A formula in a cell can “point” to a formula in another cell using an indirect reference. Cells containing these pointers will be calculated using the formulas being pointed to by the indirect reference. This indirect reference is written as \$col.row where the “\$” indicates an indirect reference and “col.row” is the cell containing the formula to be indirectly referenced.

CREATING SPREADSHEETS FROM TCL

CompuSheet⁺ provides two new verbs which can be used to create a spreadsheet from an ACCESS sentence. These verbs are SPREAD and SSPREAD. These verbs function much like the ACCESS verbs LIST and SORT, however, instead of creating a listing, a new spreadsheet is created from the data specified in the sentence. Most of the standard ACCESS conventions apply to SPREAD and SSPREAD. These include list processing, selection criteria, sort keys, output specifications and print limiters, most modifiers, and most options. Standard conversions and correlatives also apply. The structure of the verbs is:

```
SPREAD {DICT} filename {itemlist} {sellist} . .
. . {outlist} {modlist} {(options)}
SSPREAD {DICT} filename {itemlist} {seqlist} . .
. . {sellist} {outlist} {modlist} {(options)}
```

Refer to the documentation on ACCESS for detailed information on the sentence structure. After entering the sentence, the following prompts will appear:

Enter the spreadsheet file name or “/X” to quit:

Enter the name of the spreadsheet file where the generated spreadsheet is to be stored. This can be any spreadsheet file. Enter a /X to cancel the verb.

Enter the name of the spreadsheet or /X to quit:

Enter the name to be assigned to the generated spreadsheet. This name must be 15 characters or less and cannot contain any spaces. A password may follow the name (separated by a comma). If the spreadsheet name matches a spreadsheet already on file which is protected by a password, the password must be entered. Enter a <cr> to return to the spreadsheet file name prompt or /X to terminate the process.

A spreadsheet with that name already exists.
Overwrite (Y or N)?

The spreadsheet name entered matches one that already exists in the file. Enter an N or <cr> to go back one prompt or enter a Y to overwrite the existing spreadsheet with the new one.

Do you want to enter CompuSheet (Y/N or /X to exit)?

After the spreadsheet is generated, the verb will either call the spreadsheet on the screen or return to TCL. Enter a Y to call the spreadsheet, N to return to TCL or <cr> to back up to the spreadsheet name prompt.

After these prompts are completed, CompuSheet⁺ will begin to build the spreadsheet. As each row is constructed, the row number will display on the lower left of the screen.

THE CS-MENU PROCESS

CompuSheet⁺ provides a menu system which is called from TCL. Upon entering CS-MENU from TCL, the following will appear:

```
----- CompuSheet+MENU -----  
1. Initiate the terminal definition routine/command  
   synonym process  
2. Create a CompuSheet+ spreadsheet file  
3. Activate CompuSheet+ on another account  
4. Remove CompuSheet+ from another account  
5. Enter CompuSheet+  
-----
```

Please enter your selection:

The following paragraphs explain the menu choices:

1. Initiate the terminal definition routine/command synonym process

This process is used to define the characteristics of the terminal being used by CompuSheet⁺. This includes the definition of the width and depth of the terminal, the width and depth of the printer being used, the sequence of characters that define the visual attributes (reverse video, color, etc.), and the ability to redefine any CompuSheet⁺ command with any other sequence of characters (including function keys, cursor keys, and other languages). This routine is comprehensive. Please refer to the user manual for details on the use of the process.

2. Create a spreadsheet file

This process is used to define a file to be used to contain spreadsheets. Upon selection the screen will display:

Enter the name of the file to be created:

Enter any valid file name. The name must be no longer than 15 characters, cannot contain spaces, cannot be numeric, and must not duplicate any existing entry in the master dictionary. The next prompt will be:

1. Up to 25 spreadsheets
2. Up to 100 spreadsheets
3. Up to 250 spreadsheets
4. Up to 500 spreadsheets
5. Up to 1000 spreadsheets

Enter a number (from 1 to 5) indicating how many spreadsheets are likely to be stored in the file:

Enter the number that corresponds with the approximate number of spreadsheets that will be stored in the file. At this point, a request will be made to verify the entries, and if correct, CompuSheet⁺ will create the file.

3. Activate CompuSheet⁺ on another account

This selection will copy the elements needed to run CompuSheet⁺ into a selected account. The prompt will display:

Enter the account name where CompuSheet⁺ is to be activated:

Enter the appropriate account name.

4. Remove CompuSheet⁺ from another account

This selection removes the elements that allow CompuSheet⁺ to run on another account. The prompt will display:

Enter the account name where CompuSheet⁺ is to
be removed:

Enter the appropriate account name.

5. Enter CompuSheet⁺

This selection will call the spreadsheet program. It is the equivalent of entering COMPUSHEET at TCL.

Pick on the IBM AT/XT



This section is for users of the Pick System on the PC-XT, PC-AT or compatibles. The documentation included in this section is current through release 2.2. As of release 2.0, Pick now supports up to three parallel printers.

CHANGES TO DISK HANDLING

The primary change to release 2.2 is the handling of disk and offline media. By enhancing the use of buffers and ROM BIOS calls, performance of diskette handling has been dramatically improved. In 2.2, all disk sectoring formats are now supported and the 17-sector track restriction has been removed. The FORMAT command now supports 5-1/4 and 3-1/2 inch diskettes. Also, Pick now flushes at least one write-required frame to disk every ten seconds.

CHANGES TO STREAMING CARTRIDGE TAPE HANDLING

As of 2.2, Pick no longer writes an empty 512-byte block between each block of data like in earlier releases. This provides the ability to read SCT's

created by other systems to be read in Pick. At the same time, it eliminates backward compatibility with earlier versions. A new option is provided for restore. It is the "O" (for Old format) option when being supplied to the prompt for file restore device.

Pick Systems advises users *not* to write end-of-file marks at the beginning of SCT's, as this has lead to some problems where tapes are unreadable. The FILE-SAVE and ACCOUNT-SAVE PROCs have been modified not to do this anymore.

Blocks of data are now written to tape from 512-byte buffers. The number of buffers set aside for the tape buffer is determined by the SET-SCT command.

Finally, the system is able to recognize unlabeled tapes. Formerly, if the tape were not labeled the first 512-byte block of data was lost. Now, if the system senses that there is no label, it automatically positions at the beginning of the first block where it can be treated as data.

CHANGES TO TERMINAL ASSIGNMENT

Prior to release 2.0, the terminal characteristics for each port were maintained in an item called TERMTYPE in the SYSPROG BP file. As of 2.0, the terminal characteristics have been moved to attribute 2 of the port location/definition item in the DICT of the ACC file. The terminal characteristics are read in from a program called TERM-TYPE which should be placed into each accounts logon PROC. The editor is used to maintain individual port characteristics. For example, to change the terminal characteristics for line 0, you can enter

```
>ED DICT ACC 00<cr>
TOP
.L2<cr>
001 Memory Mapped Monitor
002 79,24,1,0,2,8,80,60,I
```

NEW EXTENSIONS TO PICK/BASIC

See the EXECUTE and PRINT @ statements in the section on PICK/BASIC. Four user exits have been added to provide PEEK and POKE memory read/write capabilities and IN and OUT port I/O capabilities. See the program called IO.TESTS in the BP file of the SYSPROG account for examples of how they are used. SYSTEM functions 12 through 17 have been added. See the PICK/BASIC section for detailed explanation.

NEW EXTENSIONS TO PROC

See the “T” command in the section on PROC. Two responses are now available to the PROC “IT” (Input from Tape) command. They are *C* to *continue* or *O* to *override* character. Also a new user exit has been added:

UE070 Returns the current reel number to the active input buffer

CHANGES TO THE SYSTEM DEBUGGER

The END and OFF commands have been changed.

➤ The END Command

Formerly, if the system debugger were entered from the PICK/BASIC debugger, an END would return control to the PICK/BASIC debugger. Now, it returns directly to TCL.

➤ The OFF Command

Formerly, if the system debugger were entered from the PICK/BASIC debugger, an OFF would return control to the PICK/BASIC debugger. Now, it logs off.

DOS CONSIDERATIONS

➤ About Partitions

If you intend to provide a partition for DOS, it *must* be set up prior to loading Pick. When Pick loads, it takes up the largest unused partition. Pick needs a minimum of two and a half megabytes in the partition.

➤ The COPYPICK Command

A verb called COPYPICK is now provided by Pick Systems. This verb allows the movement of Pick items into a DOS partition and must be executed from DOS. Under DOS, the COPYPICK command has the following format:

C>COPYPICK<cr>

The program then prompts for options:

- D Diagnostics. When supplied as the first option, summary diagnostics are provided. When in the second option position, for example, after the I option, more detailed diagnostics are provided
- I Includes the (Pick) item-id as a line within the DOS file
- N Numeric item-ids. Prompts for beginning and ending range numbers, and items are moved in ascending sequential order

After answering the options prompt, the system prompts with:

PICK Account name:
PICK File (DICT FILE or FILE or
FILE,DATA):
DOS File:
Pick Item or * for all:

After the copy, a count of the total number of records and attributes is displayed. See the DOS documentation for restrictions on file names.

➡➡ The FDISK Command

The DOS command, FDISK, is also relevant. The FDISK command has also been added to the Pick system and has similar operational characteristics to its DOS counterpart.

GRAPHICS SUPPORT

A new (unofficial and unsupported) user-exit is now provided with Pick for pixel-addressable graphics. It is illustrated in the program called MMVI-DEO, which can be found in the BP file of the SYSPROG account.

BOOTING THE PICK SYSTEM

There are two methods to reboot Pick on the XT/AT :

- (1) Using the REBOOT verb discussed later in this section, or,
- (2) By pressing the CTRL, ALT, and DEL keys simultaneously on port 0 (zero).

During the initial loading process, a series of percent signs (%) are displayed. In the three second gap while these appear on the monitor, and before the options prompt appears, several valid options are possible:

- A Allows changing the number of ABS frames
- F Indicates that system will recognize Streaming Cartridge Tape (SCT). (Release 2.2)
- S Toggles overlapped SCT and disk I/O processing. Increases system performance when using tape and disk simultaneously. This defaults to not enabled. (Release 2.2)
- W Toggles the ROM BIOS multi-tasking capability. This defaults to off on two-user systems and to on for systems with three or more users. (Release 2.2)

Rebooting automatically takes place from the hard disk, unless the Pick PC system diskette #1 is present in the diskette drive, in which case, the following displays on the screen:

OPTIONS:K)ill, A)BS only, F)ile & ABS, Q)uick
file & ABS

- K Delete the Pick partition
- A Used to restore the Pick monitor and ABS section only. (This is typically referred to as an "ABS load"). After loading, control transfers to the COLDSTART procedure. This reloads everything except the data files
- F Used to re-initialize the hard disk partition in which Pick is resident, followed by a restore of the monitor, ABS section and file area. (This is typically referred to as a "full restore"). See also the :FILES command
- Q Exactly like the F option, except that the disk is not initialized

FILE RESTORE OPTIONS

Formerly, the following options appeared for the file restore device:

- H High density floppy
- S Standard density floppy
- Q Quarter-inch SCT

The prompt for file restore device now includes a new option and a change to an old option:

- O Old SCT. Used when the SCT was created on a pre-2.2 release
- Q Quarter-inch SCT. Used when the SCT was created on a 2.2 or higher release

When prompted to change diskettes, two responses are valid:

- C Continue, after correct diskette has been inserted
- Q To quit for any reason

SHUTTING DOWN THE SYSTEM

See the POWER-OFF command later in this section.

CHANGES TO LOGON

A change has been made to the way the logon process is handled. After three unsuccessful logon attempts, the port is rendered “inactive”, preventing further attempts. To “re-activate” the port, press any key (other than the <cr>) followed by two <cr>'s.

ABS FRAME USAGE

The following chart shows which of the ABS frames are available for use in the Pick AT version. Note that as of 2.2, Pick now provides 704 ABS frames. Pick Systems advises assembler programmers to use frames above 550. The * indicates that this frame or group of frames is checked during the VERIFY-SYSTEM command.

<i>ABS frame(s)</i>	<i>Status</i>
001 – 109	Reserved by Pick Systems
110	Available
111 – 116	Reserved by Pick Systems
117	Available
118	Reserved by Pick Systems
119	Available
120 – 129	Reserved by Pick Systems
130	Available
131 – 184	Reserved by Pick Systems
185	Available
186 – 207	Reserved by Pick Systems
208	MAINLINK
209 – 220	Reserved by Pick Systems
221	Available
222 – 223	JET
224 – 226	Reserved by Pick Systems
227	Available
228 – 280	Reserved by Pick Systems
281	Available
282 – 284	Reserved by Pick Systems
285	Available
286 – 311	Reserved by Pick Systems

<i>ABS frame(s)</i>	<i>Status</i>
312–313	COMPU-SHEET
314–324	Reserved by Pick Systems
325–337	Available
338–339	Reserved by Pick Systems
340–380	JET
381–383	Reserved by Pick Systems
384–396	Available
397–398	Reserved by Pick Systems
399–399	ACCU-CURSOR
400–403	Reserved by Pick Systems
404	Available
405	Reserved by Pick Systems
406	Available
407	Reserved by Pick Systems
408–417	ACCU-PLOT
418–419	Reserved by Pick Systems
420–421	COMPU-SHEET
422–422	Reserved by Pick Systems
423–428	COMPU-SHEET
429	Reserved by Pick Systems
430–459	ACCU-PLOT
460–467	COMPU-SHEET
468–469	Reserved by Pick Systems
470–485	MAINLINK
486–511	COMPU-SHEET
512–550	JET
551–703	Available

PICK XT/AT TCL COMMANDS

>:FILES

Initiates a full restore as an alternative to doing it from the “options” prompt. The proper media type must be indicated with the SET-FLOPPY or SET-SCT prior to using this command. (Release 2.2)

>ADDENDA

Activates PROC to display information about Operating System enhancements made by Pick Systems.

>ADDENDUM *number*

Instructs PROC to display information about a specific feature related to Operating System enhancements made by Pick Systems. Use this after the ADDENDA command.

>BEEP

Activates PICK/BASIC program to continuously beep on a CRT until someone reaches over and slams their fist down on any of the keys.

>CAT {*filename*}

Activates PROC to display information about the compile date/time for a PICK/BASIC program file.

>CLOCK

Activates program to demonstrate operation of real-time clock.

>COLOR {*foreground*},{*background*}/{*options*}

Sets foreground and/or background colors on the color graphics monitor. The options allow special visual attributes to be activated or deactivated. May only be used from port 0 (zero).

Valid Color Choices

BLACK	GREEN
BLUE	MAGENTA
BROWN	RED
CYAN	WHITE

Valid Options for Visual Attributes

/B or /BLINK	Activate blinking
/F or /FULL	Full intensity foreground
/H or /HALF	Half intensity foreground
/NB or /NOBLINK	Deactivate blinking
/NR	Deactivate reverse video
/NOREVERSE	Deactivate reverse video
/R or /REVERSE	Activate reverse video

>COPYDOS DOS*path* {(*options*)}

TO:(*filename* {*itemname*})

Copies data, in Pick storage format, from a DOS partition into the Pick partition. To address the DOS*path*:

drive:\filename
drive:\directory\filename
drive:\directory\directory\filename

The pattern continues. Effectively there could be many directories. The options are:

- D Displays diagnostic information during translation
- F Flag characters. System prompts for further information
- L Creates list-type (indirect) item. (Release 2.2)
- M Create multiple (Pick) items
- O Overwrite (duplicate) Pick item(s)
- P Strips most significant bit off characters as they are converted, except for flagged and/or translated characters. (Release 2.2)
- R Indicates DOS data is in a random file
- S Indicates DOS data is in a sequential file
- T Translate characters. System prompts for further information
- X Creates hexadecimal image of DOS file

When file type (S or R) is omitted, the default is S. Note that when this is executed without the T and F options, DOS end-of-line characters (carriage returns or X'0D) are automatically converted to attribute marks (X'FE'). Line feeds (X'0A' and nulls X'00') are deleted.

>CS

Clears the terminal screen.

>DCD {*portnumber*}

Displays the current status of the DCD (Data Carrier Detect) on a specified port, or for the current port if one is not specifically requested. DCD is used to sense changes in the signal being transmitted over the wire to the terminal; when it drops, the terminal is automatically logged off. Note that special cabling is required prior to enabling this feature on a line. (Release 2.2)

>DCD-OFF {*portnumber*}

Disables the monitoring of the Data Carrier Detect on the specified *port-number*. If the port number is omitted, the current port is affected. May not be issued on or against line 0 (zero). The default setting is off.

>DCD-ON {*portnumber*}

Enables the monitoring of Data Carrier Detect on the specified *port-number*. If the port number is omitted, the current port is affected. May not be issued on or against line 0 (zero). The default setting is off.

>DEFINE-TERMINAL

Allows the addition or modification of term types for using different types of terminals on the system. All information needed for the process is requested from the menu presented upon issuing the command. See also the TEST-CURSOR command.

As of release 2.2, the following additional terminal characteristics have been added to DEFINE-TERMINAL.

<i>Code</i>	<i>Description</i>
@(- 11)	Activate screen protect
@(- 12)	Deactivate screen protect
@(- 13)	Activate reverse video
@(- 14)	Deactivate reverse video
@(- 15)	Activate underlining
@(- 16)	Deactivate underlining
@(- 17)	Activate slave printer
@(- 18)	Deactivate slave printer
@(- 19)	Move cursor right
@(- 20)	Move cursor down
@(- 21)	Activate graphic character set
@(- 22)	Deactivate graphic character set
@(- 23)	Keyboard lock
@(- 24)	Keyboard unlock
@(- 25)	Control character enable
@(- 26)	Control character disable
@(- 27)	Write status line
@(- 28)	Clear status line
@(- 29)	Initialize terminal mode
@(- 30)	Download function keys
@(- 31)	Non-embedded stand-out on
@(- 32)	Non-embedded stand-out off
@(- 99)	Returns a 1 if the terminal uses embedded attributes or 0 (zero) if it does not. Returns null if not defined through DEFINE-TERMINAL
@(- 100)	Half-intensity (except on memory mapped monitor)
@(- 101)	Full-intensity (except on memory mapped monitor)

>EPSON

Activates PICK/BASIC program to change characteristics for the EPSON printer.

>FC

Displays the current status of “flow control” or DSR (Data Set Ready) and X-ON/X-OFF protocol on a specified port, or for the current port if one is not specifically requested. When the DSR signal is off or the X-OFF protocol is invoked, the output to the port is suspended. (Release 2.2)

>FC-OFF {*portnumber*}

Disables the flow control on the specified port number. If the port number is omitted, the current port is affected. May not be issued on or against line 0 (zero). The default setting is on. See also the FC-ON, MODEM-ON and MODEM- OFF commands. (Release 2.2)

>FC-ON {*portnumber*}

Enables the flow control on the specified *portnumber*. If the port number is omitted, the current port is affected. May not be issued on or against line 0 (zero). The default setting is on. See also the FC-OFF, MODEM-ON and MODEM-OFF commands. (Release 2.2)

>FDISK

Activates menu to allow any of the following functions:

- Change partition pointer. This affects the boot process next time it is used
- Delete the Pick partition
- Display partition information
- Create a PICK Partition
- Select next fixed disk drive. (Release 2.2)
- Undo changes to partition data, leaving them the way they were prior to invoking FDISK. (Release 2.2)

This command may only be issued from the SYSPROG account while on port 0 (zero). See also the REBOOT command.

>FILECOMP

Activates PICK/BASIC program to compare and analyze the contents of two files. Pick Systems advises reviewing the program itself prior to using it, as it requires having a “special” file established.

>FIND

Activates PICK/BASIC program to search a file for the existence of one or more strings of characters in any attribute.

Compatibility: Pick AT/XT

>FKEYS

Activates PICK/BASIC program used for maintaining function key definition items. Pick Systems advises reviewing the program in the SYSPROG BP file prior to using it.

>FORMAT

Activates process to format floppy diskettes for use *only* with the Pick system. This command may only be issued from the SYSPROG account. Note that as of release 2.2, the FORMAT command now supports 5-1/4 and 3-1/2 inch diskettes.

>LIST-PORTS {(Z)}

Displays the baud rate, parity bits, stop bits and word length for every connected port on the system. See also the SET-PORT command. (Release 2.2) The option is:

Z Displays the status for all configured ports, connected or not.

>MODEM-OFF

Disables the flow control on the current port. May not be issued on or against line 0 (zero). The default setting is on. See also the FC, FC-ON, FC-OFF, and MODEM-ON commands. (Release 2.2)

>MODEM-ON

Enables the flow control on the current port. May not be issued on or against line 0 (zero). The default setting is on. See also the FC, FC-ON, FC-OFF, and MODEM-OFF commands. (Release 2.2)

>MONO {/options}

Sets options for the monochrome display unit. The options allow special visual attributes to be activated or deactivated. May only be used from port 0 (zero).

Valid Options for Visual Attributes

/B or /BLINK	Activate blinking
/F or /FULL	Full intensity foreground
/H or /HALF	Half intensity foreground
/NB or /NOBLINK	Deactivate blinking
/NR	Deactivate reverse video

/NU	Deactivate underlining
/NOREVERSE	Deactivate reverse video
/NOUNDERLINE	Deactivate underlining
/R or /REVERSE	Activate reverse video
/U	Activate underlining
/UNDERLINE	Activate underlining

>OKIDATA

Activates a menu to allow the changing of printer characteristics for the Okidata parallel printer.

>PACK

Activates PICK/BASIC program to compact multiple items into composite items for transmission. Pick Systems provides this program, but no support for it. See also the UNPACK command.

>POWER-OFF

Brings the system to an “orderly shutdown” by flushing memory to disk, disabling all users, and parking the disk drive heads prior to halting the system. May only be executed from port 0 (zero) when all other users are logged off. As of release 2.2, this command will not work if any lines other than zero are active. This includes spooler-controlled lines (printers).

>REBOOT

Causes the system to re-initialize, rather than using the ctrl-ALT-DEL keys. May only be issued from port 0 (zero). Disables all users and flushes memory prior to rebooting. See also the FDISK and POWER-OFF commands. As of release 2.2, this command will not work if any lines other than zero are active. This includes spooler-controlled lines (printers).

>RESTORE-ACCOUNTS

Activates PROC to restore multiple accounts from a FILE-SAVE tape or diskette(s). See also the ACCOUNT-RESTORE command. (Release 2.2)

>SET-BAUD {*portnumber*,}*baud rate*

Allows changing the baud rate for lines 1 and 2 on the XT, and for 1 through 10 on the AT. The default baud rate for each line is 9600. When the line (port) number is not specified, the current line is affected. See also the SET-PORT command. The valid baud rate choices are:

110, 150, 300, 600, 1200, 2400, 4800, 9600
(and 19200 on release 2.2 and higher)

>SET-DATE *mm/dd/yy{yy}* {(U)}

Sets the system date to the specified setting. See also the SET-DATE-EUR and SET-DATE-STD commands. Note that this is now a PROC that executes the verb called SET.DATE. The option is:

- U Updates the hardware real-time clock, as well as the system (software clock)

>SET-DATE-EUR

Toggles the standard date format to European format: *dd/mm/yy*.

>SET-DATE-STD

Toggles the European date format to North American format: *mm/dd/yy*.

>SET-FLOPPY {(*density,drive*)}

Used after the T-ATT command to designate the density to read/write diskettes and which floppy drive to use, when more than one is present. Drive choices are typically A or B. See also the FORMAT command. Don't forget to "rewind" the diskette with the T-REW command prior to attempting any read or write operation. Usually, this doesn't take too long. The defaults are: drive A, High density. As of release 2.2, 3-1/2 inch floppies are supported. The density choices are:

- S Standard density (360Kb for 5-1/4 inch, 720Kb for 3-1/2 inch)
- H High density (1.2Mb for 5-1/4 inch, 1.44Mb for 3-1/2 inch)

>SET-FUNC *filename itemname*

Specifies name of item to use in assigning function keys for use only by port 0 (zero). Each function key may be assigned one character. See the file called FUNCKEYS on the SYSPROG account. The default upon booting is that the function keys are undefined. See the item called SET-KBRD.DOC in the DOC file in the SYSPROG account for more information on defining function keys.

>SET-KBRD *filename itemname*

Specifies name of item to use in assigning the type of keyboard for use only by port 0 (zero). The items available in the KEYBOARDS file on the SYSPROG account include: BRITISH (ENGLISH), DVORAK, FRENCH, FRENCH 2, GERMAN, ITALIAN, SPANISH, SPANISH 2 and USA (the default). See the item called KBRD.DOC, in the DOC file on the SYSPROG account. It contains a detailed explanation for constructing keyboard defini-

tion items. As of release 2.2, Pick Systems now provides support for 101/102 keyboards.

>SET-LPTR

>SET-LPTR *number*

>SET-LPTR {*number*{,*delay*},{*burst*},{*slice*}}

Used to display or change performance parameters on parallel printers. To obtain an optimum blend of printer and system performance, experiment with different settings. When no parameters are provided, the current settings are displayed. A printer number may be specified to display its current settings. Like the TERM command, existing parameters may be left intact by providing two successive commas where the parameter would normally appear. The *number* parameter indicates the printer number in the range 0 to three.

The *delay* parameter indicates the number of times to attempt to send a character to the printer. The higher the value, the slower the system performance, but faster printer performance. The default is 300, and the value must be in the range 1 to 4095.

The *burst* parameter indicates the number of characters to transmit to the printer in one packet. Relates to size of printer (RAM) buffer. The default is 200, and must be in the range 1 to 255. Pick Systems suggests that 15 provides good performance.

The *slice* parameter indicates the number of timeslices that the printer waits prior to receiving its next time slice. The default is 1, and it must be in the range 1 to 15. Pick Systems advises using a number equal to or greater than the number of users on the system.

>SET-PORT {*portnumber*{,*baud*,*parity*,*stopbits*,*wordlength*}}

Activates program to display or change baud rate, parity bits, stop bits and word length for a port. When no portnumber is specified, the existing settings are displayed. The valid baud rates (as of release 2.2) are:

110, 150, 300, 600, 1200, 2400, 4800, 9600,
and 19200. The default baud rate is 9600.

To leave an existing parameter intact, use two successive commas (like the TERM command). *Parity* may be N for none, O for odd or E for even. *Stopbits* may be either 1 or 2. *Wordlength* may be either 7 or 8. See also the LIST-PORT command.

>SET-SCT (*blocksize*)

(Pick XT version.) Used after the T-ATT command to designate that the Streaming Cartridge Tape should be used for any subsequent "tape" opera-

tions. If the *blocksize* is omitted, it will default to 16384. The *blocksize* must be between 2048 and 16384, and a multiple of 512.

>SET-SCT {*buffers*} {{{*blocksize*}{*R*}}

(Pick AT version.) Used after the T-ATT command to designate that the Streaming Cartridge Tape should be used for any subsequent “tape” operations. The *buffers* parameter indicates the number of 512-byte buffers to assign for the tape buffer. The default (and minimum) is 128. The maximum is a function of how much memory is present on your system. Up to half of the memory may be allocated. For instance, if your system has 640Kb of main memory, the maximum number of buffers is 640. If the *blocksize* is omitted, it defaults to 16384. The block size must be between 512 and 32500, and be a multiple of 512. The option is:

R Indicates that the tape is in “old” format, meaning that it was created on a version prior to 2.2.

>SET-TIME *hh:mm:ss* {(U)}

Sets the system time to specified setting. Note that this is now a PROC that executes the verb called SET.TIME. The option is:

U Updates the hardware real-time clock, as well as the system (software clock)

>SETUP.SIO

This program was provided with release 2.0 to alter the parity, stop bits and word length for a port. As of release 2.2, it has been renamed SET-PORT.

>T-ERASE

Performs a re-tension and erases a Streaming Cartridge Tape.

>T-RETEN

Re-tensions Streaming Cartridge Tape by moving all the way to the end of the tape, then rewinding. Recommended prior to using tape for any purpose. Automatically performed during FILE-SAVE and ACCOUNT-SAVE.

>T-STATUS

Indicates the type and drive number of the currently attached offline storage device.

>TA {*portnumber*}

Displays status of type-ahead buffer on specified port number, or current port if omitted. See also the TA-OFF and TA-ON commands. (Release 2.2)

>TA-OFF {*portnumber*}

Disables type-ahead buffer on specified port number, or current port if omitted. See also the TA and TA-ON commands. (Release 2.2)

>TA-ON {*portnumber*}

Enables type-ahead buffer on specified port number, or current port if omitted. See also the TA and TA-OFF commands. (Release 2.2)

>TERM-TYPE

Activates PICK/BASIC program to automatically set the terminal characteristics for the current line. The item read in to set the characteristics is found in the DICT of the ACC file. The item-id is the 2-digit port number. The editor is used to maintain individual port characteristics. For example, to change the terminal characteristics for line 0, you can enter:

```
>ED DICT ACC 00<cr>
TOP
.L2<cr>
001 Memory Mapped Monitor
002 79,24,1,0,2,8,80,60,I
```

>TEST-CURSOR

Activates program to test the various terminal capabilities such as clear screen, clear to end of line, "X,Y" addressing, reverse video, etc. See also the DEFINE-TERMINAL command. (Release 2.2)

>UNPACK

Activates PICK/BASIC program to convert composite items back into the smaller, multiple items created with the PACK command. Pick Systems provides this program, but no support for it. See also the PACK command.

>WHICH

Provides current release level.

>XCS {*portnumber*}

Displays status of Extended Character Set for specified port number or current port if omitted. When XCS is enabled, the high-order bit is not

stripped from input. This allows characters in the (decimal) range 128 to 239 to be used. See also the XCS-OFF and XCS-ON commands. (Release 2.2)

>XCS-OFF {*portnumber*}

Disables extended character set on specified port number or current port if omitted. See also the XCS and XCS-ON commands. (Release 2.2)

>XCS-ON {*portnumber*}

Enables extended character set on specified port number or current port if omitted. See also the XCS and XCS-OFF commands. (Release 2.2)

Glossary

The PICK System is unique in many respects, not the least of which is its own set of terminology. In writing this guide, it was necessary to create some new terms in order to reduce the amount of redundant re-defining of functions and parameters. The following term conventions and definitions are used consistently throughout The PICK Pocket Guide:

<cr> Abbreviation for carriage return. On some keyboards, the key for a <cr> may be called “Newline”, “Linefeed”, “Return” or “Enter”.

ABS frame A frame containing the object code for the operating system. Name derived from “*Absolute location*”.

ABS section The virtual memory storage area from frames 0 through 1024, containing the object code for the operating system and all the frames set aside for user and application system assembler code.

account A collection of logically related files associated with one user or one function on a PICK system. Also the entry required to log on to the system.

acctname Abbreviation for “*account name*”, as it appears in the SYSTEM file.

AMC Abbreviation for *Attribute Mark Count*, or the relative position of an attribute within an item. Used by ACCESS to locate the attribute for retrieval.

argumentlist A list of expressions, each separated by a comma, used to designate information shared between programs using the CALL and SUBROUTINE statements from PICK/BASIC.

arrayvar Abbreviation for dynamic “*array variable*”. A variable defined for the storage of data elements in a meaningful pattern, as used in PICK/BASIC.

attrexp Abbreviation for “*attribute mark count expression*”; a PICK/BASIC expression which yields a number to be used as an attribute mark count (AMC) reference.

attribute Synonym for “field” in the PICK world. A specific piece of data which, along with other attributes, constitutes an item or “record”. The following example illustrates an item with six attributes:

001 JES & Associates, Inc.
002 P.O. Box 00000
003 Irvine
004 CA
005 92714
006 714-555-5555

attribute mark The reserved character used to separate attributes in an item. For example:

Hexadecimal Value — FE
Decimal Value — 254
ASCII Value — <control> {<shift>} ^

attrname Abbreviation for “*attribute name*”. A data element referred to by its name via an Attribute-Defining-Item in the dictionary of the file.

base fid The first “*frame-id*” in the block of (primary) space used to define a file.

baud rate The speed at which data is transmitted to or from a device. For example, CRTs are typically set to a baud rate of 9600 baud, indicating that they are to receive 9600 bits (960 bytes) per second.

bit Abbreviation for “*Binary Digi*”. The means by which data is stored on a computer. Each bit represents either a “1” or a “0” with an electrical charge.

byte Eight bits. One character of storage. For example, the character, “A”, requires one byte of storage, and is represented in the binary form, “01000001”.

buffer A (disk) ABS or data frame while it is in main memory.

col,row Abbreviation for “*column and row*”, the “X,Y” coordinates for positioning the cursor on the terminal screen in PICK/BASIC. The “column” indicates how many positions to move horizontally across the screen. The “row” indicates how many lines to move vertically down the screen.

condexp Abbreviation for “*conditional expression*”; a PICK/BASIC expression which evaluates to a zero (false) or a numeric non-zero (true).

conversion Defines an action to be taken on a specified attribute or the results of a correlative during the post-processing phase of an ACCESS process, which is just before the data is printed and/or displayed. For historical significance, the technical standard committee of the Spectrum Manufacturers Association (SMA) has dubbed a conversion as *reversible-mapping*, meaning that it can either work on an internal-to-external basis or an external- to-internal basis.

convexp Abbreviation for “*conversion expression*”; a PICK/BASIC expression which evaluates to any legal conversion (or correlative), as defined in the section on ACCESS. Note: The “A” and “F” correlatives may NOT be used in the conversion expression.

correlative Defines an action to be taken on a specified attribute or set of attributes during the pre-processing phase of an ACCESS process, which is before data is selected and/or sorted. For historical significance, the technical standard committee of the Spectrum Manufacturers Association (SMA) has dubbed a correlative as “non-reversible-mapping”, meaning that it can not be reversed from its output to its input. For instance, this would be the case of a formula that added several numbers together to produce a result.

CRT Abbreviation for Cathode Ray Tube. Refers to a terminal with a video screen. Also called Visual Display Terminal (VDT).

decnum Abbreviation for “*decimal number*”, as used in the ADDD, DIVD, MULD, and SUBD commands documented in the TCL section.

del Abbreviation for “*delimiter*”. (As used in the Editor section). Any non numeric character used in a multi-part editor command to provide a logical separator between parameters. The character chosen as the delimiter must *not* appear in the string of characters being affected by the command.

delimiter A delimiter may be either:(1) A character used to provide for separation of parameters within a command or function, as in the Editor commands and some correlatives; (2) A special character used to separate data elements in an item or string. Special characters are from X'FC'-'FE'. See appendix ASCII for definitions.

dictionary The level of the file hierarchy which contains the items that define the data in the related data section for use with ACCESS. The dictionary also contains an item (or multiple items) that provide the physical disk location for the data file (or files).

exp Abbreviation for “*expression*”; a variable, literal, constant, string, function, expression enclosed in parentheses, or a combination of variables and/or literals separated by operators used in a PICK/BASIC program.

fid Abbreviation for “*frame-id*”. The integer or hexadecimal reference to any particular frame of data on the disk.

file A collection of logically related items.

filename The specific name of a file to access. May also be a Q-pointer to any file on the account or system. Valid filename references may be any of the following four formats:

- 1) *filename*. This means the DATA section of the specified file on any file-handling command. For example:

```
>LIST CUSTOMER-FILE
```

- 2) *DICT filename*. This refers to the DICT (dictionary) level of the specified file. For example:

```
>LIST DICT CUSTOMER-FILE
```

- 3) *dictfilename,datafilename*. This format is used with files having been created with more than one data section. See the CREATE-FILE command in the TCL section. For example:

```
>LIST STAFF,EXECUTIVES
```

- 4) *DATA filename*. This unique form is used ONLY in the CREATE-FILE, DELETE-FILE and/or the CLEAR-FILE commands. For example:

```
>CLEAR-FILE DATA CUSTOMER-FILE
```

filevar Abbreviation for “*file variable expression*”; the filename variable assigned as the object of an OPEN statement in a PICK/BASIC program.

frame The basic storage area of the system. Historically, each frame is 512 bytes in length and is capable of storing 500 bytes of data, with the first 12 bytes reserved for linkage fields. *Note 1:* Some PICK systems now have data frame sizes other than 512 bytes; consult your vendor if you do not know the size of the frames on

your system. *Note 2:* All of the “software” implementations of PICK have ABS frames greater than 512; typically they are either 2K (2048 bytes) or, in some cases, 4K (4096 bytes).

GFE Abbreviation for *Group Format Error*. A condition indicating errors in the virtual storage (data) frames of a file. Before the introduction of PICK tools for correcting GFEs, this acronym was sometimes taken to mean “Gone For Ever”.

group The name given to a statistically grouped series of items stored in a PICK file. Can logically represent many frames in a linked format. The number of groups in a file is indicated by the modulo of the file.

hashing The process of passing an item-id through a mathematical algorithm to determine the primary frame of the group in which the item belongs. This represents the standard data access method used in the PICK system, sometimes referred to as a “hashed open bucket” access method.

hexnum Abbreviation for “*hexadecimal number*”, as used in the ADDX, DIVX, MULX, and SUBX commands documented in the TCL section.

idexp Abbreviation for “*item-id expression*”; a PICK/BASIC expression which evaluates to an item-id.

indexexp Abbreviation for “*index expression*”; a PICK/BASIC expression which evaluates to a number to correspond with the numeric position of a statement label in an “ON . . GOTO” or “ON . . GOSUB” statement.

item Synonym for “record” in the PICK world. Differs from a traditional record in that an item in a PICK system may contain *fields* or *attributes* that are not physically located in the item. Therefore, an *item* is a collection of logically related attributes. Items are stored in groups in a PICK system; a set of logically related groups constitutes an PICK file.

item-id Abbreviation for *Item-Identifier*. The key or unique identifier to an item in a file. May be any combination of alphanumeric characters, with no practical length restrictions. The use of certain characters should be avoided in constructing item-ids. These characters include: quotes (‘ or “), backslashes (\), spaces (), parentheses (()), “up-arrows” (^), and NEVER, EVER, ANY CONTROL CHARACTERS!

itemname Item-id or key. Refers to a single item.

itemlist (ACCESS form) The specific list of items to affect by an ACCESS process, such as a SORT, LIST, COUNT, SELECT, etc. Unless otherwise instructed, ACCESS will affect all items in a file. May be presented in any of the following formats:

- 1) A single item name, surrounded by quote marks (“”) or backslashes (\\).
- 2) Multiple item names, each surrounded by single quotes or backslashes.
- 3) The result of a SELECT, SSELECT, QSELECT or GET-LIST.

itemlist* (TCL-II form) The specific list of items to be affected by any TCL-II process, such as a COPY, ED (EDIT), CATALOG, BASIC, etc. All TCL-II commands must have this parameter indicated, and it may be presented in any of the following formats:

- 1) A single *item name*, surrounded by spaces.
- 2) Multiple *item names*, each surrounded by spaces.
- 3) The result of a SELECT, SSELECT, QSELECT or GET-LIST.
- 4) An asterisk (*), which means ALL items currently in the file.

K Abbreviation for Kilobytes. 1024 Bytes. Typically used in reference to the amount of RAM memory in a computer.

line# The port number of a particular process. Typically used with commands in which a process may be affected by its port number, such as LOGON, LOGOFF and MSG. The *line#* for a process may be displayed with the WHO, WHAT, WHERE or LISTU commands.

linkage The reserved area of each data frame used to define the forward and/or backward pointers to the attached (linked) “overflow” frames. Usually the first twelve bytes of a 512-byt.e PICK data frame is reserved for linkage te information, and the remaining 500 bytes is available for the storage of data. See also *frame*.

listname The item-id of a *list* of items, as stored in the POINTER-FILE, or an optionally specified *filename*. These “lists” are created with the SAVE-LIST, COPY-LIST, or EDIT-LIST verbs. Also used with the GET-LIST and DELETE-LIST commands.

literal An alphanumeric string of characters enclosed within quotes, or a number with or without quotes as used in a PICK/BASIC program.

MD Abbreviation for *Master Dictionary*. The primary vocabulary file for each account. The MD contains verbs, file pointers, PROCS, ACCESS attribute definition items and various modifiers and connectives. The items placed into each new account MD by the CREATE-ACCOUNT process reside in the NEWAC file in the SYSPROG account.

megabyte One million bytes. Typically used in reference to the amount of (disk) storage available on a computer, before RAM memory got large enough to be measured in megabytes.

modlist Abbreviation for “*modifier list*”. The list of modifiers used to perform special functions on output of an ACCESS report, such as ID-SUPP, DET-SUPP, LPTR, etc. See the Modifier section of ACCESS for more information.

modulo The number of groups in a file. Defined during the file creation process.

monitor The PICK Operating System software component that controls all Input, Output and process scheduling.

numexp Abbreviation for “*numeric expression*”; a PICK/BASIC expression which evaluates to a number.

outlist Abbreviation for “*output list*”. The names of the Attribute-Defining-Items to output in an ACCESS report.

options Options are typically single-character designators that appear at the end of a command line to alter the “normal” effect of a particular command. Almost without exception, options are enclosed in parentheses, although the right parenthesis is normally optional. “Standard” options are discussed at the beginning of each section of the Guide. Options that are unique to a command are discussed with the command.

overflow (or overflow table) The list of the available frames on disk that are not being used to store information. Displayed with the POVf command.

parnum Abbreviation for “*parameter number*”. Typically used as a reference to one of the locations in the PROC Primary Input Buffer. *Parnum* can alternately mean whichever input buffer location is active at the time of the instruction.

PCB Abbreviation for *Primary Control Block*. The first frame of each port/process’ workspace. Contains the process’ related accumulators, registers and counters. See the WHERE instruction in the TCL section.

PIB Abbreviation for *Process Identification Block*. Used by the monitor as a scratch-pad for process scheduling.

pointer Refers to the *line pointer* counter of the current line number in the editor item.

Primary Input Buffer An area through which data is sometimes capable of being passed from a terminal to the operating system. Typically addressed by the PROC language.

Q-pointer A type of file pointer definition that allows an account to access files defined in a different account.

segment mark The reserved character used to delimit items within a frame on disk.

Hexadecimal Value — FF

Decimal Value — 255

ASCII Value — _

selist Abbreviation for “*selection criteria list*”. The selection criteria to limit data being accessed, normally composed of one or more “WITH” clauses in an ACCESS sentence. Unless otherwise instructed, ACCESS will affect all items in a file.

separation The number of frames per group, typically used in the CREATE-FILE command. Normally, separation is “1”, since the PICK system automatically links as many frames together as it needs to store data in a group.

seqlist Abbreviation for “*sequence list*”. The collation, or sort, sequence of a report, as defined by one or more “BY” modifiers in an ACCESS sentence. The *seqlist* parameter is only available with ACCESS verbs that sort.

statement A PICK/BASIC command statement.

strexp Abbreviation for “*string expression*”; a PICK/BASIC expression which evaluates to a string of alphabetic and/or numeric characters.

subvalexp Abbreviation for “*subvalue mark count expression*”; a PICK/BASIC expression which yields a number to be used as a sub-value mark count (SVMC) reference.

sub-value A sub-portion of a value, within an attribute. Delimited by a sub-value mark. For example:

Hexadecimal Value — FC

Decimal Value — 252

ASCII Value — \

SYSTEM The primary file on the PICK system. This file defines all accounts and several system-level files that are accessible to all accounts.

TCL Abbreviation for *Terminal Control Language*. The point from which all operations begin. Indicated with a > prompt character. In some versions of PICK, there is now support for multiple levels of TCL. Each subsequent level adds an additional prompt character to remind the user where they are. For example, the second level of TCL uses >> as the prompt character.

timeslice The number of milliseconds allowed to each process upon activation by the PICK Monitor software.

valexp Abbreviation for “*value mark count expression*”; a PICK/BASIC expression which yields a number to be used as an value mark count (VMC) reference.

value A sub-portion of an attribute, delimited by a value mark. For example:

Hexadecimal Value — FD

Decimal Value — 253

ASCII Value — <control>]

variable A symbol whose alphanumeric value changes from one repetition of a program to the next in a PICK/BASIC program.

Index

A

Access, 71-107
A, 90, 91
AND, 74
arithmetic functions, 94
arithmetic operators, 94
attribute-defining-item format, 77, 89
B, 72
BREAK-ON, 84
BY, 84
BY-DSND, 84
BY-EXP, 85
BY-EXP-DSND, 85
C, 72, 90
C:attrnum, 95
Celement, 95
CHECK-SUM, 78
Cn, 98
COL-HDR-SUPP, 85
command use notes, 72
concatenation operators, 94, 98
correlatives and conversions, 89
COUNT, 78
D, 72, 90, 93, 96, 98
D:attrnum, 96
DBL-SPC, 85
DELETE-LIST, 78
DET-SUPP, 85
DICT, 85
EACH, 85
EDIT-LIST, 78
EVERY, 86
F, 72, 90, 97
FILE-TEST, 79
FOOTING, 86
FS, 100
functional operators, 99
G, 90, 101
GET-LIST, 79
GRAND-TOTAL, 86
H, 72
HASH-TEST, 79

HDR-SUPP, 86
HEADING, 87
I, 72
ID-SUPP, 87
IF, 87
ISTAT, 79
L, 90, 101
LIST, 79
LIST-ITEM, 80
LIST-LABEL, 80
logical operators, 73
LPTR, 87
LPV, 100
M, 90, 102
mathematical operators, 98
MC, 90, 103
MD, 90
ML, 90
modifiers and connectives in, 84
MP, 90
MR, 90
MS, 91, 103
MT, 91, 104
multiple string searches in statements, 76
MX, 91, 104
MY, 91, 104
N, 72
NA, 100
NB, 93, 100
ND, 93, 100
NI, 93, 100
NL, 100
NOPAGE, 87
NS, 93, 100
NV, 93, 100
ONLY, 88
OR, 74
P, 72, 91, 99, 104
QSELECT, 80
quotation use in sentences, 73
R, 91, 99, 105

REFORMAT, 81
relational operators and effects, 94, 98
S, 91, 99, 105
S-DUMP, 81
SAVE-LIST, 81
SELECT, 81
sentence format in, 72
SMA standard compatibility, 71, 91
SORT, 81
SORT-ITEM, 82
SORT-LABEL, 82
special operands, 100
special symbols and alternate meanings, 74
SREFORMAT, 82
SSELECT, 83
STAT, 83
string searching, 75
SUM, 83
SUPP, 88
T, 91, 93, 98, 105
T-DUMP, 83
T-LOAD, 83
TAPE, 88
throwaway connectives, 75
TOTAL, 88
U, 91
U0070, 107
U1070, 107
U201E, 107
U2070, 107
U3070, 107
U4070, 107
U5070, 107
U508E, 107
U6070, 107
U7070, 107
U9070, 107
UA070, 107
UC070, 107
user exits and correlatives and conversions, 106

USING, 88
V, 91
V::, 107
verbs in/, 78-83, 78
WITH, 75, 76, 89
WITHOUT, 89
Y, 72
Z, 72
arithmetic functions, Access, 94
arithmetic operators
Access, 94
CompuSheet +, 269
array reference expressions, PICK/BASIC, 27
ASCII table, 249-255
assembly format, 5

B

BASIC, PICK/BASIC, 17
Boolean expressions, PICK/BASIC, 26

C

CATALOG, PICK/BASIC, 18
CMND?, 4
COL#, 4
columns, 5
COMPILE, PICK/BASIC, 17
CompuSheet +, 257-279
=, 262
arithmetic operators, 269
AU, 262
AVG, 270
backslash, 261
C, 261
CA, 262
CALL, 275
calling PICK/BASIC subroutines
from formulas, 275
CD, 263
CL, 263

CNT, 270
 CO, 263
 COLOR, 275
 command menu for, 259
 command summary, 262
 creating spreadsheet from TCL
 SPREAD and SSPREAD, 276
 CS-menu process, 277-279
 cursor control in, 259
 D, 261, 274
 DE, 263
 DI, 263
 E, 261
 ED, 263
 editor for, 260
 entering, 257
 entering data, formulas, and headings,
 260
 EX, 264
 FB, 264
 FC, 275
 FI, 264
 FIELD, 272
 financial functions, 275
 FIND, 274
 FINDW, 274
 FL, 264
 FO, 264
 formula entry in, 268
 formula errors and, 268
 functions, 269
 GET, 273
 GETW, 273
 GR, 264
 I, 261
 ICONV, 274
 ID, 265
 IF, 265, 271
 IN, 265
 INDEX, 272
 IRR, 276
 LEN, 272
 literals, 269
 LO, 265
 MA, 265
 MAX, 271
 ME, 265
 MIN, 270
 miscellaneous commands, 274
 N, 274
 NE, 265
 NPV, 275
 O, 261
 OCONV, 275
 PA, 266
 password for, 258
 PR, 266
 precedence and precision, 268
 R, 261
 RC, 266
 READ, 272
 reading information from database
 with, 272
 READW, 273
 referencing other formula, 276
 referencing spreadsheet cells,
 columns, rows, 269
 retrieving data from other spreadsheet
 cells, 273
 RS, 266
 RU, 266
 S, 261
 SA, 266
 SO, 267
 SPACE, 272
 SPREAD, 276
 spreadsheet name for, 257
 SR, 267

SPREAD, 276
 ST, 267
 STR, 272
 string operators, 272
 SUB, 272
 SUM, 269
 T, 274
 table lookup, 274
 TE, 267
 TRIM, 272
 WI, 267
 WN, 267
 X, 262
 XE, 268
 Z, 262
 concatenation operators
 Access, 94, 98
 PICK/BASIC, 28
 correlatives and conversions, Access,
 89, 106

D
 DECATALOG, PICK/BASIC, 19
 default file variables, PICK/BASIC,
 29-30
 delimiters, Editor, 2
 dimensioned array reference, PICK/BA-
 SIC, 27
 dynamic array reference, PICK/BASIC,
 27

E
 Editor, 1-16
 command format for, 1
 Gn, 9
 Editor
 <cr>, 3, 4
 ?, 3, 4
 A, 1, 3, 5
 AS, 3, 5
 B, 3, 5
 C, 3, 5
 command format for, 2
 command summary, 4-16, 4
 DE, 2, 3, 5
 delimiters for, 2
 ED, 1
 EDIT, 1
 EDIT-LIST, 2
 error messages in, 4
 EX, 3, 6
 EXK, 3, 6
 F, 3, 6
 FD, 2, 3, 6
 FDK, 3, 6
 FI, 2, 3, 6
 FIK, 3
 FIL, 3, 7
 FIO, 3, 7
 FS, 2, 3, 7
 FSL, 3, 8
 FSO, 3, 8
 Gn, 3
 I, 3, 9
 L, 2
 Ln, 3, 9
 M, 1, 3
 ME, 2, 3, 11
 n, 3, 11
 P, 1, 11
 PD, 3, 12
 Pn, 3, 12
 R, 2, 3, 12
 RECOVER-FD, 1
 RU, 3, 12
 S, 1, 3, 14

S?, 3, 14
 SMA standards for, 2
 T, 3, 14
 TB, 3, 14
 U, 14
 wrapup error messages, 4
 X, 3, 14
 XF, 3, 15
 Z, 1, 3, 15
 Zb-e, 15
 ^, 3, 4
 EOI n, 4
 error messages
 Editor, 4
 PICK/BASIC, 241-248
 system, 215-239

F
 financial functions, CompuSheet+, 275
 functional operators, Access, 99
 functions, CompuSheet+, 269

I
 IBM AT/XT computer usage, 281-289
 :FILES, 287
 ABS frame usage, 286
 ADDENDA, 287
 ADDENDUM, 288
 BEEP, 288
 booting PICK system, 284
 CAT, 288
 CLOCK, 288
 COLOR, 288
 COPYDOS DOS, 288
 COPYPICK, 283
 CS, 289
 DCD, 289
 DCD-OFF, 289
 DCD-ON, 289
 DEFINE-TERMINAL, 290
 disk handling, 281
 DOS considerations, 283
 EPSON, 290
 extensions to PICK/BASIC, 282
 extensions to Proc, 283
 FC, 291
 FC-OFF, 291
 FC-ON, 291
 FDISK, 284, 291
 file restore options, 285
 FILECOMP, 291
 FIND, 291
 FKEYS, 292
 FORMAT, 292
 graphics support, 284
 LIST-PORTS, 292
 LOGON, 286
 MODEM-OFF, 292
 MODEM-ON, 292
 MONO, 292
 OKIDATA, 293
 PACK, 293
 POWER-OFF, 293
 REBOOT, 293
 RESTORE-ACCOUNTS, 293
 SET-BAUD, 293
 SET-DATE, 294
 SET-DATE-EUR, 294
 SET-DATE-STD, 294
 SET-FLOPPY, 294
 SET-FUNC, 294
 SET-KBRD, 294
 SET-LPTR, 295
 SET-PORT, 295
 SET-SCT, 295, 296
 SET-TIME, 296

SETUP.SIO, 296
 streaming cartridge tape handling, 281
 system debugger, 283
 system shut-down, 286
 T-ERASE, 296
 T-RETEN, 296
 T-STATUS, 296
 TA, 297
 TA-OFF, 297
 TA-ON, 297
 TCL commands, 287
 TERM-TYPE, 297
 terminal assignment, 282
 TEST-CURSOR, 297
 UNPACK, 297
 WHICH, 297
 XCS, 297
 XCS-OFF, 298
 XCS-ON, 298
 item-ids, 73

J
 Jet, 133-146
 A, 134
 ANCHOR, 138
 B, 134
 backslash asterisk command, 137
 backslash commands, 137
 BC 1, 138
 BC END, 138
 BC START, 138
 BC2, 138
 BEGIN, 138
 BEGINPAGE, 138
 BIN, 138
 BP, 138
 C, 134, 139
 CENTER, 139
 CENTER START, 139
 CH, 139
 CHAIN, 139
 CHAPTER, 139
 CRT, 139
 cursor movement, 134
 deleting text, 137
 edit mode, 136
 ENDIF, 139
 F, 134, 140
 FOOTING, 140
 FOOTING OFF, 140
 GALLEY, 140
 H, 134, 140
 HEADING, 140
 HEADING OFF, 140
 HILITE OFF, 141
 HL, 141
 HL OFF, 141
 HLITE, 141
 HLITE OFF, 141
 IFATT, 141
 IFENTER, 142
 INDENT, 142
 INDEX, 142
 INDEX.FILE, 142
 insert mode, 135
 J, 134, 143
 JET-EDIT, 133
 JET-IN, 133
 JET-OUT, 133
 JUSTIFY, 143
 key functions, 134
 L, 134
 LEN, 143
 LPI, 143
 LPTR, 143
 M, 134
 mark text commands, 136

- N, 134
 - NJ, 143
 - NOJUSTIFY, 143
 - NUMBER, 143
 - overview options, 137
 - P, 134
 - PAGE, 143
 - PAGE LENGTH, 143
 - PAUSE, 143
 - PC, 144
 - PCLOSE, 144
 - PF, 144
 - PFILE, 144
 - PI, 144
 - PITCH, 144
 - PN, 143
 - PROMPT, 144
 - Q, 134
 - R, 144
 - READ, 144
 - READNEXT, 144
 - replacing text, 135
 - ruler edit commands, 136
 - S, 134
 - SB, 144
 - SECTION, 145
 - SELECT, 145
 - SETSECTION, 145
 - SP, 145
 - SPACING, 145
 - special key commands, 135
 - T, 134
 - TCL commands, 133
 - TEST, 145
 - TEST PAGE, 145
 - text insertion commands, 137
 - TITLE, 146
 - TITLE.FILE, 146
 - TP, 145
 - V, 134
 - WINDOW, 146
 - Z, 134
- L**
- L n, 4
 - LIST-ITEM, PICK/BASIC, 19
 - literals, 269
 - locate, 10
 - logical expressions, PICK/BASIC, 26
 - logical operators, Access, 73
 - logon PROC, 109
- M**
- macros, Editor, 10
 - masking function, PICK/BASIC, 28
 - mathematical expressions, PICK/BASIC, 25
 - mathematical operators, Access, 98
 - modifiers and connectives, Access, 84
- N**
- NOT ON FILE, 4
- O**
- object pointers, PICK/BASIC, 19
- P**
- pattern matching expressions, 27
 - PICK/BASIC, 17-69
 - ", 23
 - !, 21, 24, 25
 - #, 21
 - &, 21, 25
 - (), 24
 - +, 22, 25
 - , 22, 25
 - /, 22, 25
 - <, 22, 25
 - =, 21, 24
 - >, 23, 25
 - @, 21, 24
 - A, 18
 - ABORT, 24, 31
 - ABS, 24, 31
 - ALPHA, 24, 31
 - AND, 24, 26
 - apostrophe, 23
 - array reference expressions, 27
 - ASCII, 24, 32
 - asterisk, 22, 24
 - backslash, 23
 - BEGIN, 24
 - BREAK, 24, 32
 - by date listing of object code, 20
 - C, 18
 - CALL, 24, 32
 - CASE, 24, 33
 - CAT, 24, 28, 33
 - CATALOG, 18
 - CHAIN, 24, 33
 - CHAR, 24, 33
 - CLEAR, 24, 34
 - CLEARFILE, 24, 29, 34
 - CLOSE, 24
 - COL1, 24, 34
 - COL2, 24, 34
 - colon, 20, 25
 - COM, 24, 34
 - comma reserved character, 21
 - command format, 17
 - COMMON, 24, 34
 - concatenation operators, 28
 - COS, 24, 35
 - COUNT, 24, 35
 - CRT, 24, 35
 - DATA, 24, 35
 - DATE, 24, 35
 - DCOUNT, 24, 35
 - DEBUG, 24, 35
 - DECATALOG, 19
 - default file variables, 29-30
 - DELETE, 29, 36
 - DIM, 24, 36
 - DIMENSION, 24
 - DO, 24, 36
 - DTX, 24, 36
 - E, 18
 - EBCDIC, 24, 36
 - ECHO, 24, 36, 37
 - ELSE, 24, 37
 - END, 24, 37, 69
 - END CASE, 37
 - ENTER, 24, 38
 - EQ, 24
 - EQU, 24, 38
 - EQUATE, 24, 38
 - EXECUTE, 24, 31, 38, 39
 - EXP, 24, 39
 - EXTRACT, 24, 39
 - FIELD, 24, 40
 - FOOTING, 24, 40
 - FOR, 24
 - FOR...NEXT, 41
 - FOR...NEXT...UNTIL, 41
 - FOR...NEXT...WHILE, 42
 - functions vs. statements, 31-65
 - GE, 24
 - GOSUB, 24, 42
 - GOT, 42
 - GOTO, 24
 - GT, 24
 - HEADING, 24, 42
 - I, 18
 - ICONV, 24, 31, 43
 - IF, 24, 43
 - IN, 24, 43
 - INCLUDE, 24, 44
 - INDEX, 24, 44
 - INPUT, 24, 31, 44
 - INPUTERR, 24, 45
 - INPUTNULL, 24, 45
 - INPUTTRAP, 24, 45
 - INSERT, 24, 45
 - INT, 24, 46
 - L, 18
 - LE, 24
 - LEN, 24, 46
 - LIST-ITEM, 19
 - LN, 24, 46
 - LOCATE, 24, 46
 - LOCK, 24, 47
 - LOCKED, 24
 - logical (Boolean) expressions, 26
 - LOOP, 24
 - LOOP...UNTIL, 47
 - LOOP...WHILE, 47
 - LT, 24
 - M, 18
 - masking function, 28
 - MAT, 24, 48
 - MATCH, 24, 27, 48
 - MATCHES, 24, 27, 48
 - MATREAD, 24, 29, 49
 - MATREADU, 24, 29, 49
 - MATWRITE, 24, 50
 - MATWRITEU, 24, 50
 - MOD, 25, 50
 - multiple END statements, 69
 - N, 18
 - NE, 25
 - NEXT, 25, 50
 - NOT, 25, 50
 - NULL, 25, 51
 - NUM, 25, 51
 - object pointers and, 19
 - OCONV, 25, 51
 - OFF, 25
 - ON, 25, 52
 - OPEN, 25, 29, 52
 - options for, 17
 - OR, 25, 26
 - OUT, 25, 52
 - P, 18
 - PAGE, 25, 52
 - pattern matching expressions, 27
 - precedence and mathematical expressions in, 25
 - PRECISION, 25, 53
 - PRINT, 25, 31
 - PRINT ON, 53
 - PRINT, 54-55, 53
 - PRINTER, 25, 55
 - PRINTER CLOSE, 55
 - PROCREAD, 25, 55
 - PROCWRITE, 25, 55
 - PROMPT, 25, 56
 - PWR, 25, 56
 - Q, 18
 - READ, 25, 29, 56
 - READNEXT, 25, 56
 - READT, 25, 57
 - READU, 25, 29, 57
 - READV, 25, 57
 - READVU, 58
 - relational operators, 28
 - RELEASE, 25, 29, 58
 - REM, 25, 58
 - REPEAT, 25, 59
 - REPLACE, 25, 59
 - reserved characters, 20
 - reserved words for, 24
 - RETURN, 25, 59
 - REWIND, 25, 59
 - RND, 25, 31, 60
 - RQM, 25, 60
 - RUN, 18
 - S, 18
 - SELECT, 25, 29, 60
 - semi-colon, 20
 - SEQ, 25, 60
 - SIN, 25, 60
 - SLEEP, 25, 61
 - SORT-ITEM, 19
 - source code files and, 19
 - SPACE, 25, 61
 - SQRT, 25, 61
 - statement labels, 30-31, 30
 - statements and functions, 31-65, 31
 - STEP, 25
 - STOP, 25, 61
 - STR, 25, 61
 - SUBROUTINE, 25, 62
 - substring expressions, 27
 - SYSTEM, 25, 62
 - TAN, 25, 63
 - THEN, 25
 - THEN...ELSE, 63, 67-69
 - TIME, 25, 63
 - TIMEDATE, 25, 63
 - TO, 25
 - TRIM, 25, 64
 - UNLOCK, 25, 64
 - UNTIL, 25, 64
 - user exits, 66
 - WEOF, 25, 64
 - WHILE, 25, 64
 - WRITE, 25, 29, 64
 - WRITET, 25, 65
 - WRITEU, 25, 29, 65
 - WRITEV, 25, 29, 65
 - WRITEVU, 25, 65
 - X, 18
 - XTD, 25, 65
 - Y, 18
 -], 23, 25
 -], 23, 25
 - ^, 22, 25
 - PICK/BASIC debugger, 203-208
 - \$, 205
 - ?, 205
 - activation of, 203
 - B, 205
 - B\$, 205
 - B\$=117, 205
 - BAMOUNT, 205
 - command summary, 205
 - D, 206
 - DE, 206
 - DEBUG, 206
 - E, 206
 - END, 206
 - G, 206
 - K, 206
 - L, 206
 - LP, 207
 - N, 207
 - OFF, 207
 - operators for, 204
 - P, 207
 - PC, 207
 - prompt character for, 204
 - R, 207
 - referencing variables for, 204
 - S, 207
 - SMA standards and, 203
 - symbol definition and, 203
 - T, 207

- U, 207
 - V, 208
 - Z, 208
 - PICK/BASIC error messages, 241-248
 - precedence and precision
 - CompuSheet+, 268
 - PICK/BASIC, 25
 - prestore, 11, 12
 - PROC, 1, 109-121
 - +number, 110
 - number, 111
 - A, 111
 - AT10A6, 118
 - B, 111
 - BO, 111
 - C, 111
 - commands, 110
 - D, 111
 - DICT, 110
 - F, 111
 - G, 112
 - GO, 112
 - GO A, 112
 - H, 112
 - IF, 112
 - IH, 113
 - IP, 113
 - IS, 113
 - IT, 114
 - linkages, 110
 - logon PROC, 109
 - O, 114
 - P, 114
 - P91BC, 120
 - PH, 114
 - PP, 114
 - PW, 114
 - PX, 114
 - RI, 115
 - RO, 115
 - S, 115
 - SMA standards and, 109
 - SP, 115
 - SS, 115
 - STOFF, 115
 - STON, 115
 - T, 115
 - U, 117
 - U01AD, 118
 - U01B8, 119
 - U01BC, 119
 - U11AD, 119
 - U11BC, 119
 - U218D, 119
 - U21A2, 119
 - U21AD, 119
 - U21BC, 119
 - U307A, 120
 - U318D, 120
 - U31AD, 120
 - U31BC, 120
 - U407A, 120
 - U41AD, 120
 - U41BC, 120
 - U50BB, 120
 - U51BC, 120
 - U61BC, 120
 - UA1BC, 121
 - UD070, 121
 - UE070, 121
 - user exits, 118
 - X, 121
 - programs, 1
 - prompt, PICK/BASIC debugger, 204
- R**
- referencing variables, PICK/BASIC
- debugger, 204
 - relational operators
 - Access, 94, 98
 - PICK/BASIC, 28
 - reserved characters, PICK/BASIC, 20
 - reserved words, PICK/BASIC, 24
 - RUN, PICK/BASIC, 18
 - Runoff, 123-134
 - &, 124
 - .asterisk, 124
 - .B, 125
 - .BEGIN PAGE, 124
 - .BOX, 124
 - .BOX OFF, 125
 - .BP, 124
 - .BREAK, 125
 - .C, 125
 - .CAPITALIZE SENTENCES, 125
 - .CENTER, 125
 - .CHAIN, 125
 - .CHAPTER, 125
 - .CONTENTS, 125
 - .CRT, 126
 - .CS, 125
 - .EC, 126
 - .F, 126
 - .FILL, 126
 - .FOOTING, 126
 - .HEADING, 127
 - .HLITE, 127
 - .HLITE OFF, 127
 - .I, 127
 - .IM, 128
 - .INDENT, 127
 - .INDENT MARGIN, 127
 - .INDEX, 128
 - .INPUT, 128
 - .J, 128
 - .JUSTIFY, 128
 - .LC, 128
 - .LEFT MARGIN, 128
 - .LINE LENGTH, 128
 - .LOWER CASE, 128
 - .LPTR, 128
 - .N, 129
 - .NCS, 129
 - .NF, 129
 - .NJ, 129
 - .NOCAPITALIZE SENTENCES, 129
 - .NOFILL, 129
 - .NOJUSTIFY, 129
 - .NOPAGING, 129
 - .NOPARAGRAPH, 129
 - .P, 129
 - .PAGE NUMBER, 129
 - .PAPER LENGTH, 129
 - .PARAGRAPH, 129
 - .PFILE, 130
 - .PRINT, 130
 - .PRINT INDEX, 130
 - .READ, 130
 - .READNEXT, 130
 - .SAVE INDEX, 130
 - .SECTION, 130
 - .SET TABS, 131
 - .SK, 131
 - .SKIP, 131
 - .SP, 131
 - .SPACE, 131
 - .SPACING, 131
 - .STANDARD, 131
 - .TEST PAGE, 132
 - .TP, 132
 - .UC, 132
 - .UPPER CASE, 132
 - <, 124
 - >, 124
 - @, 124
 - backslash, 124
 - C, 123
 - command format, 123
 - command summary, 124-132
 - I, 123
 - J, 123
 - n, 123
 - options in, 123
 - P, 123
 - S, 123
 - special control characters for, 124
 - U, 123
 - ^, 124
 - ____, 124
- S**
- SEQN?, 4
 - SORT-ITEM, PICK/BASIC, 19
 - source code files, PICK/BASIC and, 19
 - spooler, 209-214
 - commands, 209
 - LISTABS, 210
 - LISTPEQS, 210
 - LISTPTR, 211
 - SP-ASSIGN, 211
 - SP-CLOSE, 211
 - SP-EDIT, 212
 - SP-KILL, 212
 - SP-OPEN, 213
 - SP-STATUS, 213
 - SP-TAPEOUT, 213
 - STARTPRT, 213
 - STARTSPOOLER, 209
 - STOPPTR, 214
 - statement labels, PICK/BASIC, 30
 - string operators, CompuSheet+, 272
 - string searching, Access, 75
 - STRING?, 4
 - substring expressions, PICK/BASIC, 27
 - system debugger, 191-202
 - A, 197
 - ADDD, 197
 - address specifications, 192
 - ADDX, 197
 - B, 198
 - C, 192, 198
 - commands, 197
 - D, 198
 - data format specification for, 192
 - data reference specification for, 192
 - data window specification, 194
 - DB, 198
 - direct references, 193
 - display prompts and special functions, 196
 - DIVD, 198
 - DIVX, 198
 - DTX, 198
 - E, 198
 - END, 199
 - G, 199
 - H, 199
 - I, 192, 199
 - indirect references, 193
 - K, 199
 - L, 199
 - M, 200
 - ME, 200
 - MULD, 200
 - MULX, 200
 - N, 200
 - OFF, 200
 - offset specification, 194
 - P, 200
 - R, 200
 - SET-SYM, syntax for, 191
- SUBD**, 200
- SUBX**, 201
- symbol file definitions for, 191
- T**, 201
- term conventions and definitions for, 192
- TIME**, 201
- U**, 201
- X**, 192, 201
- XTD**, 201
- Y**, 201
- Z**, 202
- system error messages, 215-239
- A, 215
 - A(n), 215
 - D, 215
 - E, 215
 - functions, 215
 - Hstring, 216
 - L, 216
 - L(n), 216
 - R(n), 216
 - S(n), 216
 - sequence of, 216
 - standard Pick system for, 216-239
 - T, 216
 - testing, 216
 - X, 216
- T**
- terminal control language (TCL)
- :FILES, 150
 - :STARTSPOOLER, 150
 - :SWD, 150
 - :SWE, 150
 - :SWX, 151
 - :SWZ, 151
 - :TASKINIT, 151
 - ACCESS verbs in, 150
 - ACCOUNT-RESTORE, 151
 - ACCOUNT-SAVE, 151
 - ADDD, 151
 - ADDENDA, 152
 - ADDENDUM, 152
 - ADDX, 152
 - B/ADD, 152
 - B/DEL, 152
 - B/UNLOCK, 152
 - BASIC, 152
 - BATCH, 153
 - BEEP, 153
 - BLOCK-PRINT, 153
 - CAT, 153
 - CATALOG, 153
 - CHARGE-TO, 154
 - CHARGES, 154
 - CHECK-SUM, 154
 - CHOO-CHOO, 154
 - CLEAR-BASIC-LOCKS, 154
 - CLEAR-FILE, 154
 - CLOCK, 154
 - COLDSTART, 154
 - COLOR, 154
 - command summary, 150-
 - COMPARE, 155
 - COMPILE, 155
 - COPY, 156
 - COPY-LIST, 157
 - COPYDOS DOS, 157
 - COUNT, 158
 - CREATE-ACCOUNT, 158
 - CREATE-FILE, 159
 - CROSS-INDEX, 159
 - CS, 160
 - CT, 160
 - CTRL-I, 148
 - CTRL-J, 148

About the Author

Jonathan E. Sisk is president of JES & Associates of Newport Beach, Calif., a firm specializing in the Pick Operating System. He is the author of *The Pick Pocket Guide*, *The Ultimate Pocket Guide*, *The REALITY Pocket Guide*, *Exploring the Pick Operating System*, and other books on Pick-based systems. Mr. Sisk also conducts a popular series of seminars nationwide on various aspects of the Pick System and is a frequent speaker at user groups and industry trade shows.

The PICK Library Series

Edited by Jonathan E. Sisk

The following books are in the PICK Library Series:

PICK BASIC:
A Programmer's Guide
by Jonathan E. Sisk

The PICK Perspective
by Ian Jeffrey Sandler

The PICK Pocket Guide
by Jonathan E. Sisk

PICK for Professionals:
Advanced Methods and Techniques
by Harvey E. Rodstein

Other PICK® Books of Interest!

The PICK® Perspective—Ian Jeffrey Sandler

With this comprehensive guide, you'll have the expertise of one of the principal architects of the modern PICK operating system at your disposal. Ian Jeffrey Sandler helps you become a seasoned PICK veteran with his insight on:

- Writing better software
- Interfacing PICK
- Expanding PICK with DOS
- Networking PICK
- Programming faster with application generators
- Producing large-scale commercial applications

- 304 pages
- Illustrated

Book No. 3123—\$34.95 (Hardcover Only)

PICK® BASIC: A Programmer's Guide—Jonathan E. Sisk

Let Jonathan E. Sisk, the leading source of PICK documentation and training, help you become an expert PICK "translator." This book will equip you to:

- Modify report formats
- Make ad-hoc database queries
- Increase efficiency in all your PICK programs
- Write code from scratch
- Write line programs

- 320 pages
- 400 illustrations

Book No. 2845—\$29.95 (Hardcover Only)



TAB Professional and Reference Books

Division of TAB BOOKS Inc.
Blue Ridge Summit, PA 17294-0850

0389

\$19.95

ISBN 0-8306-3245-X



9 780830 632459