

**Tektronix®**  
COMMITTED TO EXCELLENCE

**TEK SPS BASIC  
V02-01  
PROGRAMMING AID**

INSTRUCTION MANUAL





**PLEASE CHECK FOR CHANGE  
INFORMATION AT THE REAR  
OF THIS MANUAL.**

**TEK SPS BASIC  
V02-01  
PROGRAMMING AID**

**Tektronix, Inc.  
P.O. Box 500  
Beaverton, Oregon 97077**


**070-2847-00  
Product Group 45**

**INSTRUCTION MANUAL**

Serial Number \_\_\_\_\_ First Printing SEP 1979  
Revised JUN 1983

Copyright © 1979 Tektronix, Inc. All rights reserved. Contents of this publication may not be reproduced in any form without the written permission of Tektronix, Inc.

Products of Tektronix, Inc. and its subsidiaries are covered by U.S. and foreign patents and/or pending patents.

TEKTRONIX, TEK, SCOPE-MOBILE, and  are registered trademarks of Tektronix, Inc. TELEQUIPMENT is a registered trademark of Tektronix U.K. Limited.

Printed in U.S.A. Specification and price change privileges are reserved.

### INSTRUMENT SERIAL NUMBERS

Each instrument has a serial number on a panel insert, tag, or stamped on the chassis. The first number or letter designates the country of manufacture. The last five digits of the serial number are assigned sequentially and are unique to each instrument. Those manufactured in the United states have six unique digits. The country of manufacture is identified as follows:

B000000	Tektronix, Inc., Beaverton, Oregon, USA
100000	Tektronix Guernsey, Ltd., Channel Islands
200000	Tektronix United Kingdom, Ltd., London
300000	Sony/Tektronix, Japan
700000	Tektronix Holland, NV, Heerenveen, The Netherlands

### RELATED TEK SPS BASIC V02 MANUALS

SYSTEM SOFTWARE  
PERIPHERAL DRIVERS  
SIGNAL PROCESSING PACKAGE  
GRAPHICS PACKAGE  
R7912 DRIVER PACKAGE  
DPO DRIVER PACKAGE  
DPO ENVELOPE COMMAND  
7912AD COMMANDS PACKAGE  
HIGH-LEVEL SUPPORT PACKAGE  
ASSEMBLY LEVEL SUPPORT PACKAGE

Although the material in this manual has been thoroughly edited and checked for accuracy, Tektronix, Inc., makes no guarantees against typographical or human errors. Also, Tektronix, Inc., assumes no responsibility or liability, consequential or otherwise, of any kind arising from misinterpretation or misuse of the material in this manual. The contents of this manual are subject to change without notice.

## TABLE OF CONTENTS

Drivers & .LDA Files .....	2
Functions .....	3
Operators .....	3
Common File Name Extensions .....	4
Guide to Notation .....	4
System Software Commands .....	5
IEEE 488 Interface Driver Commands .....	44
Signal Processing Commands .....	51
Graphics Commands .....	54
R7912 Driver Commands .....	62
DPO Envelope Command .....	66
7912AD Commands .....	67
High-Level Support Commands .....	71
Assembly Level Support Commands .....	77
Useful Formulas .....	78
Abbreviated Error Codes .....	79
Booting the System .....	83
CP4165 ODT Commands .....	83
System Control Characters .....	84
ASCII & IEEE 488 (GPIB) Code Chart .....	84
Index .....	85



## DRIVERS & .LDA FILES

### Resident Monitor/Interpreter Files

name:	device:
SPSDK.LDA	hard disk
SPSDX.LDA	floppy disk

### Instrument Driver Modules

name:	device:
GPI.SPS	CP4100/IEEE 488 Interface or CP1100/IEEE 488 Interface
DPO.SPS	Digitizing Oscilloscope
TD.SPS	R7912 Transient Digitizer

### Peripheral Device Driver Modules

name:	device:
DK.SPS	hard disk
DX.SPS	floppy disk
CT.SPS	cassette tape
PR.SPS	paper-tape reader
PP.SPS	paper-tape punch
LP.SPS	line printer
MT.SPS	9-track tape
VM.SPS	virtual memory
CLK.SPS	clock



## FUNCTIONS

Function (Argument*)	Returns
ABS(X)	Absolute value of X.
ASC(A\$)	ASCII code value of first character in A\$.
ATN(X)	Arctangent of X; result is in range of $\pm\pi/2$ radians.
CAN(A\$)	Waveform units cancellation.
CHR(N)	One-character string with ASCII value N.
COS(X)	Cosine of X where X is in radians.
CRS(Y,N)	First interpolated subscript position at which elements of Y cross value N.
EXP(X)	e raised to X power.
ITP(X)	Integer part of X.
LEN(A\$)	Number of characters in A\$.
LOG(X)	Natural logarithm of X.
MAX(Y)	Maximum single value in Y.
MEA(Y)	Mean value of Y.
MIN(Y)	Minimum single value in Y.
POS(A\$,A\$,N)	Position of second string within first string. starting comparison at position N.
RMS(Y)	Root mean square value of Y.
RND(X)	Random number between 0 and 1.
SEG(A\$,N,N)	Specified part of A\$.
SGN(X)	Sign of X; 1 if $X>0$ , 0 if $X=0$ , -1 if $X<0$ .
SIN(X)	Sine of X when X is in radians.
SIZ(Y)	Total number of elements in Y.
SQR(X)	Square root of X.
STR(N)	String containing decimal value of N as if printed.
TRM(A\$)	A\$ without trailing blanks.

## FUNCTIONS (cont.)

TSK(X)	Task number of currently executing command.
VAL(A\$)	Numeric value of A\$.

### \*Argument key

N may be any constant or numeric expression.  
 X may be any constant, numeric expression, array expression, or waveform expression.  
 Y may be any array, waveform, or zone of array.  
 A\$ may be any string or string expression.

## OPERATORS

Symbol	Example	Meaning
ARITHMETIC		
↑ or ^	X↑Y	Exponentiation, 1st priority
*	X*Y	Multiplication, 2nd priority
/	X/Y	Division, 2nd priority
+	X+Y	Addition, 3rd priority
-	X-Y	Subtraction, 3rd priority
RELATIONAL		
=	X=Y	Equal to
<	X<Y	Less than
<= or <=	X<=Y	Less than or equal to
>	X>Y	Greater than
>= or >=	X>=Y	Greater than or equal to
<> or <>	X<>Y	Not equal to
STRING		
&	VA\$&VB\$	Concatenate VB\$ to VA\$



## COMMON FILE NAME EXTENSIONS

.BAS	BASIC program
.SPS	Software module
.OVL	System overlay file
.LDA	Program in Loader format
.SAV	Program in Save format
.BOL	Fast overlay file
.DAT	Data file
.FIX	Patch file
.NRS	Patch file
.BLD	Patch file

## GUIDE TO NOTATION

Syntax describes how commands may be entered to the system. Uppercase characters and punctuation must be typed as shown. Brackets [] enclose optional information. Items enclosed in braces {} must be entered into the command. When more than one item appears between braces, only one item may be selected. Items followed by three periods may be repeated. For example:

```
RELEASE { ALL  
         string expression [,string expression]... }
```

In this command, either the word ALL or a string expression must be entered. If a string expression is entered, it may be followed by a comma and another string expression. Additional commas and string expressions may also be entered.

PLUNs and ILUNs may be entered as constants, variables, or expressions. File names may be strings, string variables, elements of string arrays, or string expressions.



## SYSTEM SOFTWARE COMMANDS

### ABORT

Terminates a single task.

#### Syntax Form:

[line no.] **ABORT** [**TASK** expression]

#### Descriptive Form:

[line no.] **ABORT** [**TASK** task number]

### ATAN2

Performs double-argument arctangent.

#### Syntax Form:

[line no.] **ATAN2**  $\left\{ \begin{array}{l} \text{floating-point variable} \\ \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\}, \left\{ \begin{array}{l} \text{floating-point variable} \\ \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\},$

$\left\{ \begin{array}{l} \text{floating-point variable} \\ \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\}$

#### Descriptive Form:

[line no.] **ATAN2** real source data, imaginary source data,  
target for arctangent of imaginary/real

### ATTACH

Allows communication with an instrument.

#### Syntax Form:

[line no.] **ATTACH** #expression **AS** device name[constant]:

#### Descriptive Form:

[line no.] **ATTACH** #ilun **AS** device name[hardware unit number]:

**BOOT**

Reloads system software from a peripheral device.

**Syntax Form:**

$$[\text{line no.}] \text{BOOT} \left[ \begin{matrix} \text{DK} \\ \text{DX} \end{matrix} \right] [\text{constant}]:$$
**Descriptive Form:**

$$[\text{line no.}] \text{BOOT} \left[ \begin{matrix} \text{DK} \\ \text{DX} \end{matrix} \right] [\text{drive number}]:$$
**CANCEL**

Removes specified files from a peripheral storage device.

**Syntax Form:**

$$[\text{line no.}] \text{CANCEL} [\text{device name}[\text{constant}]:] \left[ / \begin{matrix} \text{F} \\ \text{R} \end{matrix} \right] [, ] \text{string expression}$$

[,string expression]...

**Descriptive Form:**

[line no.] **CANCEL** [device name[drive number]:][ /forward or reverse switch[,]]  
file name [,file name]...

**CHAIN**

Deletes the current program and loads and starts executing the specified new program. Does not delete variables.

**Syntax Form:**

$$[\text{line no.}] \text{CHAIN} \left[ \text{device name}[\text{constant}]: \left[ / \begin{matrix} \text{F} \\ \text{R} \end{matrix} \right] [, ] \right] [\text{string expression}][, \text{expression}]$$
**Descriptive Form:**

[line no.] **CHAIN** [device name[drive number]:][ /forward or reverse switch[,]]  
[program file name][,line number where execution continues]



**CHANGE**

Edits program text in memory.

**Syntax Form:**

[line no.] **CHANGE** [expression[,expression],]string expression[,string expression][,DEL]

**Descriptive Form:**

[line no.] **CHANGE** [line number[starting, line number ending],] text to be deleted  
[,text to be inserted][,DELeTe to end of line switch]

**CLEAR**

Initializes all variables and arrays to zero, string variables to null strings.

**Syntax Form:**

[line no.] **CLEAR**

**CLOSE**

Terminates I/O with a device or file.

**Syntax Form:**

[line no.] **CLOSE** { #expression  
ALL }

**Descriptive Form:**

[line no.] **CLOSE** { #plun  
ALL pluns }

## COPY

Transfers file from one peripheral device to another device or file.

### Syntax Form:

[line no.] **COPY** [device name[constant]:]  $\left[ \begin{array}{c} \text{F} \\ \text{R} \end{array} \right] [, ]$  [string expression]

**TO** [device name[constant]:][string expression][ **INTO** expression]

### Descriptive Form:

[line no.] **COPY** [device name[drive number]:][ /forward or reverse switch[,]]  
[source file name] **TO** [device name[drive number]:][target file name]  
[ **INTO** number of blocks].

## DATE

Obtains system date.

### Syntax Form:

[line no.] **DATE**  $\left[ \begin{array}{l} \text{simple numeric variable} \\ \text{array} \\ \text{string variable} \end{array} \right]$

### Descriptive Form:

[line no.] **DATE**  $\left[ \begin{array}{l} \text{target variable} \\ \text{target array} \\ \text{target string variable} \end{array} \right]$



**DEFINE**

Creates a Record I/O file.

**Syntax Form:**

[line no.] **DEFINE** [device name[constant]:]string expression

**AS**  $\left\{ \begin{array}{l} \text{VAR} \\ \text{ARR expression} \\ \text{STG expression} \end{array} \right\} \left[ , \left\{ \begin{array}{l} \text{VAR} \\ \text{ARR expression} \\ \text{STG expression} \end{array} \right\} \right] \dots \text{WITH expression}$

**Descriptive Form:**

[line no.] **DEFINE** [device name[drive number]:]file name

**AS**  $\left\{ \begin{array}{l} \text{VARiable} \\ \text{ARRay number of floating-point array elements} \\ \text{STrinG number of characters in string} \end{array} \right\}$

$\left[ , \left\{ \begin{array}{l} \text{VARiable} \\ \text{ARRay number of floating-point array elements} \\ \text{STrinG number of characters in string} \end{array} \right\} \right] \dots$

**WITH** number of records

## DELETE

Removes program lines, waveforms, arrays, and string arrays from memory.

### Syntax Form:

$$[\text{line no.}] \text{ DELETE } \left\{ \begin{array}{l} \text{array} \\ \text{waveform} \\ \text{string array} \\ \text{line number[,line number]} \\ \text{TEXT} \\ \text{ALL} \end{array} \right\} \left[ \begin{array}{l} \left\{ \begin{array}{l} \text{array} \\ \text{waveform} \\ \text{string array} \\ \text{line number[,line number]} \\ \text{TEXT} \end{array} \right\} \dots \end{array} \right]$$

### Descriptive Form:

$$[\text{line no.}] \text{ DELETE } \left\{ \begin{array}{l} \text{array} \\ \text{waveform} \\ \text{string array} \\ \text{line number[starting,line number ending]} \\ \text{all program TEXT in memory} \\ \text{ALL program text and data in memory} \end{array} \right\}$$
$$\left[ \begin{array}{l} \left\{ \begin{array}{l} \text{array} \\ \text{waveform} \\ \text{string array} \\ \text{line number[starting,line number ending]} \\ \text{all program TEXT in memory} \end{array} \right\} \dots \end{array} \right]$$



## DETACH

Terminates communication with an instrument.

### Syntax Form:

[line no.] DETACH { #expression  
ALL }

### Descriptive Form:

[line no.] DETACH { #ilun  
ALL iluns }

**DIM**

Assigns floating-point storage space for array variables or defines string arrays.

**Syntax Form:**

[line no.] **DIM**  $\left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{simple string variable} \\ \text{floating-point array} \\ \text{string array} \end{array} \right\} (\text{expression}[, \text{expression}])$

$\left[ \begin{array}{l} \left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{simple string variable} \\ \text{floating-point array} \\ \text{string array} \end{array} \right\} (\text{expression}[, \text{expression}]) \end{array} \right] \dots$

**Descriptive Form:**

[line no.] **DIM**  $\left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{simple string variable} \\ \text{floating-point array} \\ \text{string array} \end{array} \right\} (\text{first dimension} [, \text{second dimension}])$

$\left[ \begin{array}{l} \left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{simple string variable} \\ \text{floating-point array} \\ \text{string array} \end{array} \right\} (\text{first dimension} [, \text{second dimension}]) \end{array} \right] \dots$



<b>DIR</b>	<p>Prints on terminal or specified device a list of files stored on a peripheral device.</p> <p><b>Syntax Form:</b></p> <p>[line no.] <b>DIR</b> [device name[constant]:][string expression] [<b>WITH BLOCK</b>] [<b>TO</b> [device name[constant]:][string expression]]</p> <p><b>Descriptive Form:</b></p> <p>[line no.] <b>DIR</b> [device name[drive number]:][file name for wild card specification] [ <b>WITH</b> starting <b>BLOCK</b> numbers printed][ <b>TO</b> [device name[drive number]:] [file name to receive directory information]]</p>
<b>END</b>	<p>Terminates all program execution, closes all files, disables instrument interrupts, returns to idle mode.</p> <p><b>Syntax Form:</b></p> <p>[line no.] <b>END</b></p>
<b>EOF</b>	<p>Designates a program line to receive program control when data from a peripheral file is exhausted.</p> <p><b>Syntax Form:</b></p> <p>[line no.] <b>EOF</b> #expression <b>GOTO</b> line number</p> <p><b>Descriptive Form:</b></p> <p>[line no.] <b>EOF</b> #plun <b>GOTO</b> line number</p>
<b>FOR</b>	<p>Specifies start of program loop and controlling parameters.</p> <p><b>Syntax Form:</b></p> <p>[line no.] <b>FOR</b> simple numeric variable = expression <b>TO</b> expression[ <b>STEP</b> expression]</p> <p><b>Descriptive Form:</b></p> <p>[line no.] <b>FOR</b> index = initial value <b>TO</b> limit [ <b>STEP</b> increment]</p>

## FORMAT

Formats a CP110 cartridge disk (a Digital Equipment Corp. cartridge disk or its equivalent).

### Syntax Form:

[line no.] **FORMAT DK**[constant]:  $\left[ \begin{array}{l} \text{expression} \\ \text{expression, VER} \\ \text{VER} \end{array} \right]$

### Descriptive Form:

[line no.] **FORMAT DK**[drive number]:  $\left[ \begin{array}{l} \text{number of directory segments} \\ \text{number of directory segments, VERify} \\ \text{VERify} \end{array} \right]$

## GET

Fetches data or status information from an instrument and stores it in specified variables or in a specified peripheral file.

### Syntax Form:

$$[\text{line no.}] \text{ GET } \left\{ \begin{array}{l} \# \text{expression} \\ \text{variable} \\ \text{array} \\ \text{waveform} \\ \text{string variable} \end{array} \right\} \left[ \begin{array}{l} \text{variable} \\ \text{array} \\ \text{waveform} \\ \text{string variable} \end{array} \right] \dots \left. \vphantom{\begin{array}{l} \# \text{expression} \\ \text{variable} \\ \text{array} \\ \text{waveform} \\ \text{string variable} \end{array}} \right\}$$
$$\text{FROM } \left\{ \begin{array}{l} \# \text{expression} [, \text{string expression}] \dots \\ @ \text{expression}, \text{expression} [, \text{expression}] \end{array} \right\}$$

### Descriptive Form:

$$[\text{line no.}] \text{ GET } \left\{ \begin{array}{l} \# \text{target plun to receive data} \\ \text{target variable} \\ \text{target array} \\ \text{target waveform} \\ \text{target string variable} \end{array} \right\} \left[ \begin{array}{l} \text{target variable} \\ \text{target array} \\ \text{target waveform} \\ \text{target string variable} \end{array} \right] \dots \left. \vphantom{\begin{array}{l} \# \text{target plun to receive data} \\ \text{target variable} \\ \text{target array} \\ \text{target waveform} \\ \text{target string variable} \end{array}} \right\}$$
$$\text{FROM } \left\{ \begin{array}{l} \# \text{source ilun } [, \text{driver-dependent specification of data or status} \\ \text{information to be obtained from instrument}] \dots \\ @ \text{IEEE 488 interface number, talk address} [, \text{secondary address}] \end{array} \right\}$$



**GETBLK**

Obtains the contents of a block from a directory-structured device.

**Syntax Form:**

[line no.] **GETBLK** [device name[constant]:][string expression,]  
expression,  $\left\{ \begin{array}{l} \text{string variable} \\ \text{array} \end{array} \right\}$

**Descriptive Form:**

[line no.] **GETBLK** [device name[drive number]:][file name,]  
block number,  $\left\{ \begin{array}{l} \text{target string variable} \\ \text{target array} \end{array} \right\}$

**GETFREE**

Obtains the amount of free memory currently available.

**Syntax Form:**

[line no.] **GETFRE** variable

**Descriptive Form:**

[line no.] **GETFREE** target variable

**GETLINE**

Obtains the line number of the line being executed.

**Syntax Form:**

[line no.] **GETLIN** variable

**Descriptive Form:**

[line no.] **GETLINE** target variable

**GETLOC**

Obtains the contents of a specified memory location.

**Syntax Form:**

[line no.] **GETLOC**  $\left\{ \begin{array}{l} \text{expression} \\ \text{string expression} \end{array} \right\}, \text{floating-point variable}[, \text{expression}, \text{expression}]$

**Descriptive Form:**

[line no.] **GETLOC**  $\left\{ \begin{array}{l} \text{decimal address} \\ \text{octal address} \end{array} \right\}, \text{target variable for contents of address}$

[,low-order bit number for obtaining segment of contents,  
high-order bit number]

**GETPRIORITY** Obtains priority of task being executed.

**Syntax Form:**

[line no.] **GETPRI** variable

**Descriptive Form:**

[line no.] **GETPRIORITY** target variable

## GOSUB

Transfers program control to a subroutine or to one of a list of subroutines.

### Syntax Form:

[line no.] **GOSUB** { line number  
expression **OF** line number[,line number] ... }

### Descriptive Form:

[line no.] **GOSUB** { line number  
line number selector **OF** line number list }

## GOTO

Transfers program control to a specified line, or to one of a list of specified lines.

### Syntax Form:

[line no.] **GOTO** { line number  
expression **OF** line number[,line number] ... }

### Descriptive Form:

[line no.] **GOTO** { line number  
line number selector **OF** line number list }



## HOOK

Writes system bootstrap program on specified peripheral device.

### Syntax Form:

[line no.] **HOOK**  $\left[ \begin{array}{c} \text{DK} \\ \text{DX} \end{array} \right] [\text{constant}]: [\text{FOR RT11 string expression}]$

### Descriptive Form:

[line no.] **HOOK**  $\left[ \begin{array}{c} \text{DK} \\ \text{DX} \end{array} \right] [\text{drive number}]: [\text{FOR RT11 file name}]$

## HOOKQ

Installs an absolute loader for .LDA files on a disk.

### Syntax Form:

[line no.] **HOOKQ**  $\left[ \begin{array}{c} \text{DK} \\ \text{DX} \end{array} \right] [\text{constant}]:$

### Descriptive Form:

[line no.] **HOOKQ**  $\left[ \begin{array}{c} \text{DK} \\ \text{DX} \end{array} \right] [\text{drive number}]:$

**IF**

Conditionally transfers control or executes another command.

**Syntax Form:**

$$[\text{line no.}] \text{ IF } \left\{ \begin{array}{l} \text{expression} \\ \text{string expression} \end{array} \right\} \text{ relational operator } \left\{ \begin{array}{l} \text{expression} \\ \text{string expression} \end{array} \right\}$$

$$\text{ THEN } \left\{ \begin{array}{l} \text{statement} \\ \text{line number} \end{array} \right\}$$
**IGNORE**

Prohibits change of program flow by specified instrument conditions.

**Syntax Form:**

$$[\text{line no.}] \text{ IGNORE } \left\{ \begin{array}{l} \left\{ \begin{array}{l} \# \text{ expression} \\ @ \text{ expression} \end{array} \right\} \left[ \left\{ \begin{array}{l} \text{TASK expression} \\ \text{string expression} \end{array} \right\} \right] \\ \text{TASK expression} \\ \text{ALL} \end{array} \right\}$$
**Descriptive Form:**

$$[\text{line no.}] \text{ IGNORE } \left\{ \begin{array}{l} \left\{ \begin{array}{l} \# \text{ ilun} \\ @ \text{ IEEE 488 interface number} \end{array} \right\} \left[ \left\{ \begin{array}{l} \text{TASK task number} \\ \text{driver-dependent interrupt} \\ \text{condition specification} \end{array} \right\} \right] \\ \text{TASK task number} \\ \text{ALL interrupt conditions for all iluns} \end{array} \right\}$$

**INPREQ**

Permits unsolicited input of data from the keyboard while a program is running.

**Syntax Form:**

$$[\text{line no.}] \text{ INPREQ } \left[ \begin{array}{l} \text{CHAR} \\ \text{NOECHO} \\ \text{CHAR, NOECHO} \end{array} \right] \text{ GOSUB line number}$$
**Descriptive Form:**

$$[\text{line no.}] \text{ INPREQ } \left[ \begin{array}{l} \text{CHARacter} \\ \text{NOECHO} \\ \text{CHARacter, NOECHO} \end{array} \right] \text{ GOSUB line number}$$
**INPUT**

Obtains ASCII values for variables from the keyboard or other peripheral device or an ASCII file, and if the variables are numeric, translates those values to binary form.

**Syntax Form:**

$$[\text{line no.}] \text{ INPUT } [\# \text{expression},] \left\{ \begin{array}{l} \text{variable} \\ \text{array} \\ \text{waveform} \\ \text{string variable} \end{array} \right\} \left[ , \left\{ \begin{array}{l} \text{variable} \\ \text{array} \\ \text{waveform} \\ \text{string variable} \end{array} \right\} \right] \dots$$
**Descriptive Form:**

$$[\text{line no.}] \text{ INPUT } [\# \text{source plun},] \left\{ \begin{array}{l} \text{target variable} \\ \text{target array} \\ \text{target waveform} \\ \text{target string variable} \end{array} \right\} \left[ , \left\{ \begin{array}{l} \text{target variable} \\ \text{target array} \\ \text{target waveform} \\ \text{target string variable} \end{array} \right\} \right] \dots$$



## INTEGER

Allocates integer-format storage for arrays.

### Syntax Form:

$$[\text{line no.}] \text{ INTEGER } \left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{integer array} \end{array} \right\} (\text{expression}[, \text{expression}])$$
$$\left[ \left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{integer array} \end{array} \right\} (\text{expression}[, \text{expression}]) \right] \dots$$

### Descriptive Form:

$$[\text{line no.}] \text{ INTEGER } \left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{integer array} \end{array} \right\} (\text{first dimension}[, \text{second dimension}])$$
$$\left[ \left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{integer array} \end{array} \right\} (\text{first dimension}[, \text{second dimension}]) \right] \dots$$

**LET**

Assigns the value of an expression to a variable, array, waveform, or string variable.

**Syntax Form:**

$$[\text{line no.}] [\text{LET}] \left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{array} \\ \text{waveform} \end{array} \right\} = \left\{ \begin{array}{l} \text{expression} \\ \text{array expression} \\ \text{waveform expression} \end{array} \right\}$$

$$\left\{ \begin{array}{l} \text{array element} = \text{expression} \\ \text{string variable} = \text{string expression} \end{array} \right\}$$
**Descriptive Form:**

$$[\text{line no.}] [\text{LET}] \left\{ \begin{array}{l} \text{target simple numeric variable} \\ \text{target array} \\ \text{target waveform} \end{array} \right\} = \left\{ \begin{array}{l} \text{expression} \\ \text{array expression} \\ \text{waveform expression} \end{array} \right\}$$

$$\left\{ \begin{array}{l} \text{target array element} = \text{expression} \\ \text{target string variable} = \text{string expression} \end{array} \right\}$$
**LIST**

Prints all or part of the current program text on the system terminal or specified peripheral device.

**Syntax Form:**

$$[\text{line no.}] \text{LIST} \left[ \begin{array}{l} \text{expression[,expression]} \\ \text{device name[constant]:[string expression][,expression[,expression]]} \end{array} \right]$$
**Descriptive Form:**

$$[\text{line no.}] \text{LIST} \left[ \begin{array}{l} \text{line number [starting,line number ending]} \\ \text{device name [drive number]:[target file name]} \\ \text{[,line number[starting,line number ending]]} \end{array} \right]$$

**LISTVAR** Lists on terminal or specified device the names and dimensions of all arrays, waveforms, variables, string variables, and string arrays currently defined.

**Syntax Form:**

[line no.] **LISTVA** [device name[constant]:[string expression]]

**Descriptive Form:**

[line no.] **LISTVAR** [device name[drive number]:[target file name]]

**LOAD** Loads specified drivers or commands into memory.

**Syntax Form:**

[line no.] **LOAD** [device name[constant]:]  $\left[ \begin{matrix} F \\ R \end{matrix} \right] [, ]$  string expression  
[,string expression] ...

**Descriptive Form:**

[line no.] **LOAD** [device name[drive number]:][ / forward or reverse switch [,]]  
driver or command name [,driver or command name] ...

**LOCKKB** Limits system input to Control-P while a program is running.

**Syntax Form:**

[line no.] **LOCKKB** [OPEN]

**MATCH** Obtains the index of the string array element containing the search string.

**Syntax Form:**

[line no.] **MATCH** string array,string expression,variable[,variable]

**Descriptive Form:**

[line no.] **MATCH** string array, search string, target variable for array index  
[,target variable for search string's starting position]



**NEXT**

Terminates FOR loop.

**Syntax Form:**

[line no.] **NEXT** simple numeric variable

**Descriptive Form:**

[line no.] **NEXT** index

**OLD**

Loads a new program or program segment into memory, deleting all existing text and variables.

**Syntax Form:**

[line no.] **OLD** [device name[constant]:]  $\left[ \begin{array}{c} / \\ \left\{ \begin{array}{c} F \\ R \end{array} \right\} \end{array} \right] [, ]$  [string expression][,line number]

**Descriptive Form:**

[line no.] **OLD** [device name[drive number]:][ / forward or reverse switch [,]]  
[program file name][,line number where execution starts]

**ONERR**

Allows processing of errors in a BASIC program.

**Syntax Form:**

[line no.] **ONERR**  $\left[ \begin{array}{l} \left\{ \begin{array}{c} \text{integer array} \\ \text{variable} \end{array} \right\} \text{GOTO line number} \\ \text{RETURN [GOTO line number]} \\ \text{NOWARN} \end{array} \right]$

**Descriptive Form:**

[line no.] **ONERR**  $\left[ \begin{array}{l} \text{target for error information GOTO line number} \\ \text{RETURN [GOTO line number]} \\ \text{NOWARNING error messages} \end{array} \right]$

## OPEN

Allows access to an existing data file, a new data file, or a non-file-structured peripheral device.

### Syntax Form:

[line no.] **OPEN** #expression **AS** [device name[constant]:]  $\left[ \left/ \begin{matrix} F \\ R \end{matrix} \right\} [, ] \right]$  [string expression]  
  
**FOR**  $\left\{ \begin{matrix} \text{READ} \\ \text{WRITE [WITH expression] [INTO expression]} \\ \text{UPDATE} \end{matrix} \right\}$

### Descriptive Form:

[line no.] **OPEN** #plun **AS** [device name[drive number]:]  
[/forward or reverse switch[,]][file name]  
  
**FOR**  $\left\{ \begin{matrix} \text{READ} \\ \text{WRITE [WITH number of buffers] [INTO number of blocks]} \\ \text{UPDATE} \end{matrix} \right\}$

## OVERLAY

Loads a new program or program segment into memory without affecting variables. Overlays lines with matching line numbers, but does not delete other program text in memory.

### Syntax Form:

[line no.] **OVERLA** [device name[constant]:]  $\left[ \left/ \begin{matrix} F \\ R \end{matrix} \right\} [, ] \right]$  [string expression]

### Descriptive Form:

[line no.] **OVERLAY** [device name[drive number]:][ / forward or reverse switch [, ]]  
[program file name]

**OVLOAD**

Performs a fast overlay of a pre-translated BASIC program segment from a file created by an OVLSAV statement.

**Syntax Form:**

[line no.] **OVLOAD** [device name[constant]:]  $\left[ \left\{ \begin{matrix} F \\ R \end{matrix} \right\} [, ] \right]$  string expression

**Descriptive Form:**

[line no.] **OVLOAD** [device name[drive number]:][ / forward or reverse switch [,]]  
file name of pretranslated text

**OVLSAV**

Creates a file containing a pre-translated BASIC program segment.

**Syntax Form:**

[line no.] **OVLSAV** [device name[constant]:] string expression[, expression[, expression]]

**Descriptive Form:**

[line no.] **OVLSAV** [device name[drive number]:] target program file name  
[, line number[starting, line number ending]]



## PRINT

Outputs ASCII information to the terminal or other peripheral device or a data file.

### Syntax Form:

[line no.] **PRINT** [#expression,] 

[expression array expression waveform expression string expression string array <b>TAB</b> (expression)	{, ;}	[expression array expression waveform expression string expression string array <b>TAB</b> (expression)]	...
--	----------	---	-----

### Descriptive Form:

[line no.] **PRINT** [#target plun,] 

[expression array expression waveform expression string expression string array <b>TAB</b> (column number)]	{, ;}	[expression array expression waveform expression string expression string array <b>TAB</b> (column number)]	...
--	----------	--	-----

## PRIORITY

Changes the priority of a running program.

### Syntax Form:

[line no.] **PRIORI** expression

### Descriptive Form:

[line no.] **PRIORITY** level

## PUT

Sends data or status information from memory to a specified instrument.

### Syntax Form:

$$[\text{line no.}] \text{ PUT } \left\{ \begin{array}{l} \text{expression} \\ \text{array expression} \\ \text{waveform expression} \\ \text{string expression} \end{array} \right\} \left[ \begin{array}{l} \left\{ \begin{array}{l} \text{expression} \\ \text{array expression} \\ \text{waveform expression} \\ \text{string expression} \end{array} \right\} \\ , \left\{ \begin{array}{l} \text{expression} \\ \text{array expression} \\ \text{waveform expression} \\ \text{string expression} \end{array} \right\} \end{array} \right] \dots$$
$$\text{INTO } \left\{ \begin{array}{l} \# \text{expression}[, \text{string expression}] \dots \\ @ \text{expression}, \left\{ \begin{array}{l} \text{array expression} \\ \text{expression}[, \text{expression}] \end{array} \right\} \left[ ; \left\{ \begin{array}{l} \text{array expression} \\ \text{expression}[, \text{expression}] \end{array} \right\} \right] \dots \end{array} \right\}$$

### Descriptive Form:

$$[\text{line no.}] \text{ PUT } \left\{ \begin{array}{l} \text{source expression} \\ \text{source array expression} \\ \text{source waveform expression} \\ \text{source string expression} \end{array} \right\} \left[ \begin{array}{l} \left\{ \begin{array}{l} \text{source expression} \\ \text{source array expression} \\ \text{source waveform expression} \\ \text{source string expression} \end{array} \right\} \\ , \left\{ \begin{array}{l} \text{source expression} \\ \text{source array expression} \\ \text{source waveform expression} \\ \text{source string expression} \end{array} \right\} \end{array} \right] \dots$$
$$\text{INTO } \left\{ \begin{array}{l} \# \text{target ilun}[, \text{driver-dependent specification of data} \\ \text{or status information to be sent to instrument}] \dots \\ @ \text{IEEE 488 interface number,} \\ \left\{ \begin{array}{l} \text{listen and secondary address pairs} \\ \text{listen address}[, \text{secondary address}] \end{array} \right\} \\ \left[ ; \left\{ \begin{array}{l} \text{listen and secondary address pairs} \\ \text{listen address}[, \text{secondary address}] \end{array} \right\} \right] \dots \end{array} \right\}$$

**PUTBLK**

Stores a physical block of data on a directory-structured device.

**Syntax Form:**

[line no.] **PUTBLK** [device name[constant]:][string expression,]  
 expression,  $\left\{ \begin{array}{l} \text{string expression} \\ \text{array expression} \end{array} \right\}$

**Descriptive Form:**

[line no.] **PUTBLK** [device name[drive number]:][file name,]  
 target block number,  $\left\{ \begin{array}{l} \text{source string expression} \\ \text{source array expression} \end{array} \right\}$

**PUTLOC**

Assigns a specified value to a memory location.

**Syntax Form:**

[line no.] **PUTLOC**  $\left\{ \begin{array}{l} \text{expression} \\ \text{string expression} \end{array} \right\}, \left\{ \begin{array}{l} \text{expression} \\ \text{string expression} \end{array} \right\}$

**Descriptive Form:**

[line no.] **PUTLOC**  $\left\{ \begin{array}{l} \text{decimal address} \\ \text{octal address} \end{array} \right\}, \left\{ \begin{array}{l} \text{decimal value to be stored at address} \\ \text{octal value to be stored at address} \end{array} \right\}$



**RANDOM**

Sets seed value of the random-number generator or returns seed value.

**Syntax Form:**

[line no.] **RANDOM** floating-point variable, floating-point variable

**Descriptive Form:**

[line no.] **RANDOM** high-order part of seed, low-order part of seed

**READ**

Obtains formatted binary and ASCII values for variables from a peripheral device or file.

**Syntax Form:**

[line no.] **READ #expression,**  $\left\{ \begin{array}{l} \text{variable} \\ \text{array} \\ \text{waveform} \\ \text{string variable} \end{array} \right\} \left[ \left\{ \begin{array}{l} \text{variable} \\ \text{array} \\ \text{waveform} \\ \text{string variable} \end{array} \right\} \right] \dots$

**Descriptive Form:**

[line no.] **READ #source plun,**  $\left\{ \begin{array}{l} \text{target variable} \\ \text{target array} \\ \text{target waveform} \\ \text{target string variable} \end{array} \right\} \left[ \left\{ \begin{array}{l} \text{target variable} \\ \text{target array} \\ \text{target waveform} \\ \text{target string variable} \end{array} \right\} \right] \dots$

## READU

Obtains unformatted binary and ASCII values for variables from a peripheral device or file.

### Syntax Form:

$$[\text{line no.}] \text{ READU } \# \text{expression} [\langle \text{expression} \rangle], \left\{ \begin{array}{l} \text{variable} \\ \text{array} \\ \text{string variable} = \text{expression} \end{array} \right\}$$
$$\left[ \left\{ \begin{array}{l} \text{variable} \\ \text{array} \\ \text{string variable} = \text{expression} \end{array} \right\} \right] \dots$$

### Descriptive Form:

[line no.] READU #source plun [⟨record number⟩],

$$\left\{ \begin{array}{l} \text{target variable} \\ \text{target array} \\ \text{target string variable} = \text{number of characters in string} \end{array} \right\}$$
$$\left[ \left\{ \begin{array}{l} \text{target variable} \\ \text{target array} \\ \text{target string variable} = \text{number of characters in string} \end{array} \right\} \right] \dots$$

**RELEASE** Removes nonresident commands or drivers from memory.

**Syntax Form:**

[line no.] **RELEASE** { string expression[,string expression] ... }  
ALL

**Descriptive Form:**

[line no.] **RELEASE** { driver or command name [,driver or command name] ... }  
ALL drivers and nonresident commands in memory

**REM** Allows inclusion of remarks in program listing.

**Syntax Form:**

[line no.] **REM** text

**RENAME** Changes the name of a file on a directory-structured device.

**Syntax Form:**

[line no.] **RENAME** [device name[constant]:]string expression **TO** string expression

**Descriptive Form:**

[line no.] **RENAME** [device name[drive number]:]old file name **TO** new file name

**RENUM** Assigns new sequential line numbers to part or all of program text in memory.

**Syntax Form:**

[line no.] **RENUM** [expression[,expression]] [**TO** expression] [**STEP** expression]

**Descriptive Form:**

[line no.] **RENUM** [line number[starting,line number ending]]  
[**TO** new line number starting] [**STEP** increment]



**REPLACE**

Replaces specified file on a peripheral device with program text currently in memory.

**Syntax Form:**

$$[\text{line no.}] \text{ REPLAC } [\text{device name}[\text{constant}]:] \left[ \left\{ \begin{array}{c} \text{F} \\ \text{R} \end{array} \right\} [, ] \right] \text{ string expression} \\ [, \text{expression} [, \text{expression}]]$$
**Descriptive Form:**

$[\text{line no.}] \text{ REPLACE } [\text{device name}[\text{drive number}]:][/\text{forward or reverse switch } [,]]$   
 program file name  $[, \text{line number}[\text{starting}, \text{line number ending}]]$

**RESCHEDULE** Puts either the current task or the task on Scheduler stack back on Scheduler queue.

**Syntax Form:**

$[\text{line no.}] \text{ RESCHE } [\text{STACK}] [\text{WITH expression}]$

**Descriptive Form:**

$[\text{line no.}] \text{ RESCHEDULE } [\text{STACK}] [\text{WITH priority level}]$

**RESET**

Resets a file that is OPEN FOR READ to the beginning of that file.

**Syntax Form:**

$[\text{line no.}] \text{ RESET } \# \text{expression}$

**Descriptive Form:**

$[\text{line no.}] \text{ RESET } \# \text{plun}$

**RETURN**

Terminates the execution of a subroutine.

**Syntax Form:**

$[\text{line no.}] \text{ RETURN}$

**REWIND**

Rewinds serial tape devices.

**Syntax Form:**

[line no.] **REWIND** device name[constant]:

**Descriptive Form:**

[line no.] **REWIND** device name[drive number]:

**RUN**

Starts program at line having lowest line number in memory.

**Syntax Form:**

**RUN** [AS TASK expression]

**Descriptive Form:**

**RUN** [AS TASK task number]

**SAVE**

Stores program lines on a specified peripheral device.

**Syntax Form:**

[line no.] **SAVE** [device name[constant]:][string expression][,expression[,expression]]

**Descriptive Form:**

[line no.] **SAVE** [device name[drive number]:][program file name]  
[,line number[starting,line number ending]]

**SCHEDULE** Queues a subroutine for execution at a specified time or after a specified time lapse.

**Syntax Form:**

[line no.] **SCHEDU**  $\left[ \begin{array}{l} \text{AFTER expression} \\ \text{AT string expression} \end{array} \right] [ \text{WITH expression} ] [ \text{AS TASK expression} ]$

**GOSUB** line number

**Descriptive Form:**

[line no.] **SCHEDULE**  $\left[ \begin{array}{l} \text{AFTER number of seconds} \\ \text{AT time specification} \end{array} \right] [ \text{WITH priority level} ] [ \text{AS TASK task number} ]$

**GOSUB** line number

**SETDATE** Sets the system date.

**Syntax Form:**

[line no.] **SETDAT** string expression

**Descriptive Form:**

[line no.] **SETDATE** date specification

**SETTIME** Sets the system time.

**Syntax Form:**

[line no.] **SETTIM** [string expression]

**Descriptive Form:**

[line no.] **SETTIME** [time specification]



**SQUISH**

Compacts files on a disk storage device.

**Syntax Form:**

[line no.] **SQUISH** device name[constant]:[**TO** device name[constant]:]  $\left[ \begin{array}{l} \text{VER} \\ \text{string expression} \end{array} \right]$

**Descriptive Form:**

[line no.] **SQUISH** source device name[drive number]:[**TO** target device name[drive number]:]  
[,bad block **VER**ification switch]

**STATUS**

Prints the current status of the system on the terminal or specified peripheral device.

**Syntax Form:**

[line no.] **STATUS**  $\left[ \begin{array}{l} \text{device name[constant]:[string expression][,SCHED]} \\ \text{string expression[,SCHED]} \\ \text{SCHED} \end{array} \right]$

**Descriptive Form:**

[line no.] **STATUS**  $\left[ \begin{array}{l} \text{device name[drive number]:[target file name][,SCHEDuler information flag]} \\ \text{target file name[,SCHEDuler information flag]} \\ \text{SCHEDuler information flag} \end{array} \right]$

**STOP**

Terminates program execution, disables instrument interrupts, and returns to idle mode.

**Syntax Form:**

[line no.] **STOP**

**SYSBLD**

Defines the contents of file to set system parameters at initialization time.

**Syntax Form:**

[line no.] **SYSBLD**

**TIME**

Obtains system time.

**Syntax Form:**

[line no.] **TIME**  $\left[ \begin{array}{l} \text{simple numeric variable} \\ \text{array} \\ \text{string variable} \end{array} \right]$

**Descriptive Form:**

[line no.] **TIME**  $\left[ \begin{array}{l} \text{target variable} \\ \text{target array} \\ \text{target string variable} \end{array} \right]$

**UNSCHEDULE** Cancels the actions of a SCHEDULE command if the specified time has not elapsed.

**Syntax Form:**

[line no.] **UNSCHE**  $\left\{ \begin{array}{l} \text{GOSUB line number} \\ \text{TASK expression} \\ \text{ALL} \end{array} \right\}$

**Descriptive Form:**

[line no.] **UNSCHEDULE**  $\left\{ \begin{array}{l} \text{GOSUB line number} \\ \text{TASK task number} \\ \text{ALL scheduled line numbers} \end{array} \right\}$

**VARTST**

Tests for set bits.

**Syntax Form:**

[line no.] **VARTST**  $\left\{ \begin{array}{l} \text{expression} \\ \text{string expression} \end{array} \right\}, \left\{ \begin{array}{l} \text{expression} \\ \text{string expression} \end{array} \right\}, \text{variable}$

**Descriptive Form:**

[line no.] **VARTST**  $\left\{ \begin{array}{l} \text{decimal value to be tested} \\ \text{octal value to be tested} \end{array} \right\},$

$\left\{ \begin{array}{l} \text{decimal specification of bits to be tested} \\ \text{octal specification of bits to be tested} \end{array} \right\}, \text{target for test result}$

**VERSION**

Obtains the version and release numbers of a nonresident command or driver.

**Syntax Form:**

[line no.] **VERSIO** [device name[constant]:]  $\left[ \begin{array}{c} \left\{ \begin{array}{c} F \\ R \end{array} \right\} \\ \left[ \begin{array}{c} F \\ R \end{array} \right] \end{array} \right] \text{string expression}[, \text{string variable}]$

**Descriptive Form:**

[line no.] **VERSION** [device name[drive number]:][ /forward or reverse switch[,]]  
driver or command name [,target string variable]



**WAIT**

Stops execution of a program until a keyboard interrupt is received or a specified amount of time has elapsed.

**Syntax Form:**

[line no.] **WAIT** [expression]

**Descriptive Form:**

[line no.] **WAIT** [number of milliseconds]

**WAVEFORM**

Associates a data array with its data sampling interval and units.

**Syntax Form:**

[line no.] **WAVEFORM**  $\left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{waveform} \end{array} \right\} \text{ IS}$

$\left\{ \begin{array}{l} \text{simple numeric variable}(\text{expression}[,\text{expression}]) \\ \text{array}[(\text{expression}[,\text{expression}])] \end{array} \right\},$

numeric variable, simple string variable, simple string variable

**Descriptive Form:**

[line no.] **WAVEFORM**  $\left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{waveform} \end{array} \right\} \text{ IS}$

$\left\{ \begin{array}{l} \text{simple numeric variable}(\text{first dimension}[,\text{second dimension}]) \\ \text{array}[(\text{first dimension}[,\text{second dimension}])] \end{array} \right\},$

data sampling interval, horizontal units, vertical units

**WHEN**

Allows specified instrument conditions to change program flow.

**Syntax Form:**

[line no.] **WHEN**  $\left\{ \begin{array}{l} \# \\ @ \end{array} \right\}$  expression [**HAS** string expression] [**AT** expression]

[**AS TASK** expression] **GOSUB** line number

**Descriptive Form:**

[line no.] **WHEN**  $\left\{ \begin{array}{l} \#ilun \\ @IEEE\ 488\ interface\ number \end{array} \right\}$  [**HAS** driver-dependent interrupt specification]

[**AT** priority level] [**AS TASK** task number] **GOSUB** line number

**WRITE**

Outputs data in formatted binary and ASCII form to a peripheral device or file.

**Syntax Form:**

[line no.] **WRITE** #expression,  $\left\{ \begin{array}{l} expression \\ array\ expression \\ waveform\ expression \\ string\ expression \end{array} \right\} \left[ \begin{array}{l} expression \\ array\ expression \\ waveform\ expression \\ string\ expression \end{array} \right] \dots$

**Descriptive Form:**

[line no.] **WRITE** #target plun,  $\left\{ \begin{array}{l} expression \\ array\ expression \\ waveform\ expression \\ string\ expression \end{array} \right\} \left[ \begin{array}{l} expression \\ array\ expression \\ waveform\ expression \\ string\ expression \end{array} \right] \dots$

**WRITEU**

Transfers unformatted binary and ASCII data to peripheral device or file.

**Syntax Form:**

$$[\text{line no.}] \text{WRITEU } \# \text{expression} [ \langle \text{expression} \rangle ], \left\{ \begin{array}{l} \text{expression} \\ \text{array expression} \\ \text{string expression} = \text{expression} \end{array} \right\}$$

$$\left[ \left\{ \begin{array}{l} \text{expression} \\ \text{array expression} \\ \text{string expression} = \text{expression} \end{array} \right\} \right] \dots$$
**Descriptive Form:**

$[\text{line no.}] \text{WRITEU } \# \text{target plun } [ \langle \text{record number} \rangle ],$

$$\left\{ \begin{array}{l} \text{expression} \\ \text{array expression} \\ \text{string expression} = \text{number of characters in string} \end{array} \right\}$$

$$\left[ \left\{ \begin{array}{l} \text{expression} \\ \text{array expression} \\ \text{string expression} = \text{number of characters in string} \end{array} \right\} \right] \dots$$



## ZERO

Initializes the specified file-structured peripheral device.

### Syntax Form:

[line no.] **ZERO** device name[constant]:  $\left[ \begin{array}{c} \text{expression} \\ \left[ \begin{array}{c} \text{F} \\ \text{R} \end{array} \right] [, ] \end{array} \right] \text{[string expression]}$

### Descriptive Form:

[line no.] **ZERO** device name[drive number]:

$\left[ \begin{array}{l} \text{number of directory segments} \\ \text{[/forward or reverse switch[,]] [file name at which to start zeroing tape]} \end{array} \right]$

## IEEE 488 INTERFACE DRIVER COMMANDS

**GETSTA** Gets the status byte of a bus-connected device.

**Syntax Form:**

[line no.] **GETSTA** @expression,variable,expression[,expression]

**Descriptive Form:**

[line no.] **GETSTA** @IEEE 488 interface number, target variable for status byte,  
talk address[,secondary address]

**GIFES** Gets the error status of the specified interface.

**Syntax Form:**

[line no.] **GIFES** @expression,variable

**Descriptive Form:**

[line no.] **GIFES** @IEEE 488 interface number,target variable

**IFDTM** Sets the data transfer mode of the specified interface.

**Syntax Form:**

[line no.] **IFDTM** @expression,string expression[,string expression] ...

**Descriptive Form:**

[line no.] **IFDTM** @IEEE 488 interface number, specification of mode of data transfer  
[,specification of mode of data transfer] ...

## POLL

Performs a serial poll of the bus-connected devices.

### Syntax Form:

[line no.] **POLL** @expression,variable,variable,variable  $\left[ \left\{ \begin{array}{l} \text{array expression} \\ \text{expression[,expression]} \end{array} \right\} \right] \dots$

### Descriptive Form:

[line no.] **POLL** @IEEE 488 interface number, target variable for status byte,  
target variable for primary address, target variable for secondary address

$\left[ \left\{ \begin{array}{l} \text{talk and secondary address pairs} \\ \text{talk address[,secondary address]} \end{array} \right\} \right] \dots$

## PPOLL

Performs a parallel poll of the bus-connected devices.

### Syntax Form:

[line no.] **PPOLL** @expression,variable

### Descriptive Form:

[line no.] **PPOLL** @IEEE 488 interface number, target variable



**RASCII**

Reads ASCII data from a bus-connected device and stores it in the specified variable(s).

**Syntax Form:**

$$[\text{line no.}] \text{ RASCII } \left\{ \begin{array}{l} \text{variable} \\ \text{array} \\ \text{string variable} \end{array} \right\} \left[ \left\{ \begin{array}{l} \text{variable} \\ \text{array} \\ \text{string variable} \end{array} \right\} \right] \dots$$

**FROM** @expression[,expression[,expression]]

**Descriptive Form:**

$$[\text{line no.}] \text{ RASCII } \left\{ \begin{array}{l} \text{target variable} \\ \text{target array} \\ \text{target string variable} \end{array} \right\} \left[ \left\{ \begin{array}{l} \text{target variable} \\ \text{target array} \\ \text{target string variable} \end{array} \right\} \right] \dots$$

**FROM** @IEEE 488 interface number[,talk address[,secondary address]]

**RBYTE**

Reads a single byte of data through an IEEE 488 interface into a numeric variable.

**Syntax Form:**

[line no.] **RBYTE** @expression,variable

**Descriptive Form:**

[line no.] **RBYTE** @IEEE 488 interface number, target variable

**SIFCOM**

Sends IEEE 488 addressed and universal commands to bus-connected devices.

**Syntax Form:**

$$[\text{line no.}] \text{ SIFCOM } @ \text{expression}, \left\{ \begin{array}{l} \text{expression} \\ \text{array expression} \\ \text{string expression} \end{array} \right\} \left[ \begin{array}{l} \text{expression} \\ \text{array expression} \\ \text{string expression} \end{array} \right] \dots$$
**Descriptive Form:**

$$[\text{line no.}] \text{ SIFCOM } @ \text{IEEE 488 interface number}, \left\{ \begin{array}{l} \text{source expression} \\ \text{source array expression} \\ \text{interface command specification} \end{array} \right\}$$

$$\left[ \begin{array}{l} \text{source expression} \\ \text{source array expression} \\ \text{interface command specification} \end{array} \right] \dots$$
**SIFLIN**

Controls the IEEE 488 interface lines.

**Syntax Form:**

$[\text{line no.}] \text{ SIFLIN } @ \text{expression}, \text{string expression} [, \text{string expression}] \dots$

**Descriptive Form:**

$[\text{line no.}] \text{ SIFLIN } @ \text{IEEE 488 interface number},$   
specification of how interface line is to be set  
 $[, \text{specification of how interface line is to be set}] \dots$

**SIFTO**

Sets the interface time-out value.

**Syntax Form:**

[line no.] **SIFTO** @expression,expression

**Descriptive Form:**

[line no.] **SIFTO** @IEEE 488 interface number, time-out value in milliseconds

**STERMC**

Designates the termination character string for ASCII data read into a string variable by a GET or RASCII statement.

**Syntax Form:**

[line no.] **STERMC** @expression,string expression

**Descriptive Form:**

[line no.] **STERMC** @IEEE 488 interface number, specification of termination character(s)

**TIFL**

Reads the current setting of the control lines of an IEEE 488 bus.

**Syntax Form:**

[line no.] **TIFL** @expression,variable

**Descriptive Form:**

[line no.] **TIFL** @IEEE 488 interface number, target variable



**WASCII**

Sends ASCII data to a bus-connected device. Converts numeric data to an ASCII string before sending it.

**Syntax Form:**

$$\begin{aligned}
 &[\text{line no.}] \text{ WASCII } \left\{ \begin{array}{l} \text{expression} \\ \text{array expression} \\ \text{string expression} \end{array} \right\} \left[ \begin{array}{l} \{ \text{expression} \} \\ \{ \text{array expression} \} \\ \{ \text{string expression} \} \end{array} \right] \dots [;] \\
 &\text{INTO } @ \text{expression} \left[ , \left\{ \begin{array}{l} \text{array expression} \\ \text{expression}[, \text{expression}] \end{array} \right\} \right] \\
 &\left[ \left\{ \begin{array}{l} \text{array expression} \\ \text{expression}[, \text{expression}] \end{array} \right\} \right] \dots
 \end{aligned}$$
**Descriptive Form:**

$$\begin{aligned}
 &[\text{line no.}] \text{ WASCII } \left\{ \begin{array}{l} \text{source expression} \\ \text{source array expression} \\ \text{source string expression} \end{array} \right\} \left[ \begin{array}{l} \{ \text{source expression} \} \\ \{ \text{source array expression} \} \\ \{ \text{source string expression} \} \end{array} \right] \dots [;] \\
 &\text{INTO } @ \text{IEEE 488 interface number} \left[ , \left\{ \begin{array}{l} \text{listen and secondary address pairs} \\ \text{listen address}[, \text{secondary address}] \end{array} \right\} \right] \\
 &\left[ \left\{ \begin{array}{l} \text{listen and secondary address pairs} \\ \text{listen address}[, \text{secondary address}] \end{array} \right\} \right] \dots
 \end{aligned}$$

**WBYTE**

Sends a byte of data through an IEEE 488 interface to the bus.

**Syntax Form:**

$$[\text{line no.}] \text{ WBYTE } @ \text{expression}, \left\{ \begin{array}{l} \text{expression} \\ \text{array expression} \end{array} \right\} \left[ \left\{ \begin{array}{l} \text{expression} \\ \text{array expression} \end{array} \right\} \right] \dots [,]$$
**Descriptive Form:**

$$[\text{line no.}] \text{ WBYTE } @ \text{IEEE 488 interface number}, \left\{ \begin{array}{l} \text{source expression} \\ \text{source array expression} \end{array} \right\} \\ \left[ \left\{ \begin{array}{l} \text{source expression} \\ \text{source array expression} \end{array} \right\} \right] \dots [,]$$

## SIGNAL PROCESSING COMMANDS

### CONVL

Performs discrete convolution operation.

#### Syntax Form:

$$[\text{line no.}] \text{ CONVL } \left\{ \begin{array}{l} \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\}, \left\{ \begin{array}{l} \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\},$$
$$\left\{ \begin{array}{l} \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\} \left[ , \left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{floating-point array} \end{array} \right\} \right]$$

#### Descriptive Form:

[line no.] **CONVL** source data, source data, target for convolved result [,cosine table]

### CORR

Performs discrete auto or cross correlation operation.

#### Syntax Form:

$$[\text{line no.}] \text{ CORR } \left\{ \begin{array}{l} \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\}, \left\{ \begin{array}{l} \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\},$$
$$\left\{ \begin{array}{l} \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\} \left[ , \left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{floating-point array} \end{array} \right\} \right]$$

#### Descriptive Form:

[line no.] **CORR** source data, source data, target for correlated result [,cosine table]



**DIFF**

Differentiates an array or waveform.

**Syntax Form:**

$$[\text{line no.}] \text{ DIFF } \left\{ \begin{array}{l} \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\}, \left\{ \begin{array}{l} \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\}$$

$$\left[ , \left\{ \begin{array}{l} \text{FOR} \\ \text{string expression} \end{array} \right\} \right]$$
**Descriptive Form:**

[line no.] **DIFF** source data, target for differentiated result [,forward difference switch]

**INT**

Integrates an array or waveform.

**Syntax Form:**

$$[\text{line no.}] \text{ INT } \left\{ \begin{array}{l} \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\}, \left\{ \begin{array}{l} \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\}$$
**Descriptive Form:**

[line no.] **INT** source data, target for integrated result

**POLAR**

Performs a rectangular-to-polar conversion.

**Syntax Form:**

$$[\text{line no.}] \text{ POLAR } \left\{ \begin{array}{l} \text{floating-point variable} \\ \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\}, \left\{ \begin{array}{l} \text{floating-point variable} \\ \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\}$$

[,expression]

**Descriptive Form:**

[line no.] **POLAR** real source data and target for magnitude component,  
imaginary source data and target for phase component [,delay estimate]

**RFFT**

Performs a real fast Fourier transform or inverse Fourier transform via a power-of-two algorithm.

**Syntax Form:**

$$[\text{line no.}] \text{ RFFT } \left\{ \begin{array}{l} \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\}, \left\{ \begin{array}{l} \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\},$$

$$\left\{ \begin{array}{l} \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\} \left[ , \left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{floating-point array} \end{array} \right\} \right]$$

$$\left[ , \left\{ \begin{array}{l} \text{DIR} \\ \text{INV} \\ \text{string expression} \end{array} \right\} \right]$$
**Descriptive Form:**

$[\text{line no.}] \text{ RFFT}$  time domain data, real component of frequency domain data,  
imaginary component of frequency domain data [,cosine table]  
[,direct or inverse transform switch]

**RFFT1**

Performs single argument fast Fourier transform or inverse Fourier transform via a power-of-two algorithm.

**Syntax Form:**

$$[\text{line no.}] \text{ RFFT1 } \left\{ \begin{array}{l} \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\} \left[ , \left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{floating-point array} \end{array} \right\} \right]$$

$$\left[ , \left\{ \begin{array}{l} \text{DIR} \\ \text{INV} \\ \text{string expression} \end{array} \right\} \right]$$
**Descriptive Form:**

$[\text{line no.}] \text{ RFFT1}$  time domain data or frequency domain data [,cosine table]  
[,direct or inverse transform switch]

## GRAPHICS COMMANDS

### DISPLAY

Plots an array or waveform into current graphics window without displaying any graticule, axes, or labels.

#### Syntax Form:

$$[\text{line no.}] \text{DISPLA} [\text{expression}[, \text{expression}],] \left\{ \begin{array}{l} \text{array expression} \\ \text{waveform expression} \end{array} \right\}$$
$$\left[ , [\text{expression}[, \text{expression}],] \left\{ \begin{array}{l} \text{array expression} \\ \text{waveform expression} \end{array} \right\} \right] \dots$$

#### Descriptive Form:

$$[\text{line no.}] \text{DISPLAY} [\text{line type}[, \text{horizontal interval}],] \left\{ \begin{array}{l} \text{array expression} \\ \text{waveform expression} \end{array} \right\}$$
$$\left[ , [\text{line type}[, \text{horizontal interval}],] \left\{ \begin{array}{l} \text{array expression} \\ \text{waveform expression} \end{array} \right\} \right] \dots$$

### DRAW

Draws a line from current pointer position to position specified in user coordinates.

#### Syntax Form:

$[\text{line no.}] \text{DRAW} \text{expression}, \text{expression}[, \text{expression}, \text{expression}] \dots$

#### Descriptive Form:

$[\text{line no.}] \text{DRAW} \text{ x coordinate in user units, y coordinate in user units}$   
 $[\text{, x coordinate in user units, y coordinate in user units}] \dots$



**DRAWON** Transfers graphic output to any device OPEN FOR WRITE.

**Syntax Form:**

[line no.] **DRAWON** #expression

**Descriptive Form:**

[line no.] **DRAWON** #target plun

**GIN** Inputs position of crosshair cursor in user coordinates.

**Syntax Form:**

[line no.] **GIN** string variable,variable,variable

**Descriptive Form:**

[line no.] **GIN** target for input character,  
target for x coordinate of crosshairs in user units,  
target for y coordinate of crosshairs in user units

**GRAPH** Plots arrays or waveforms complete with graticule and axis labels.

**Syntax Form:**

[line no.] **GRAPH** { array expression[,array expression] ...  
waveform expression[,waveform expression] ... }

**INITG** Initializes graphics device (erases screen on graphics terminal) and resets window and viewport to default values.

**Syntax Form:**

[line no.] **INITG**

**MOVE** Moves pointer to position specified in user coordinates. (No line is drawn.)

**Syntax Form:**  
 [line no.] **MOVE** expression,expression

**Descriptive Form:**  
 [line no.] **MOVE** x coordinate in user units, y coordinate in user units

**PAGE** Erases screen or graphics terminal.

**Syntax Form:**  
 [line no.] **PAGE**

**RDRAW** Draws a line from current pointer position to a point displaced from current position by specified number of user units.

**Syntax Form:**  
 [line no.] **RDRAW** expression,expression[,expression,expression] ...

**Descriptive Form:**  
 [line no.] **RDRAW** x displacement in user units, y displacement in user units  
 [,x displacement in user units, y displacement in user units] ...

**RESETG** Initializes window and viewport to default values.

**Syntax Form:**  
 [line no.] **RESETG**

**RMOVE** Moves pointer to position displaced from current position by specified number of user units.

**Syntax Form:**  
 [line no.] **RMOVE** expression,expression

**Descriptive Form:**  
 [line no.] **RMOVE** x displacement in user units, y displacement in user units

**RSDRAW** Draws line from current pointer position to a point displaced from current position by specified number of graphic device units.

**Syntax Form:**  
[*line no.*] **RSDRAW** *expression*,*expression*[,*expression*,*expression*] ...

**Descriptive Form:**  
[*line no.*] **RSDRAW** *x displacement in device units*, *y displacement in device units*  
[,*x displacement in device units*, *y displacement in device units*] ...

**RSMOVE** Moves pointer to position displaced from current position by specified number of graphic device units.

**Syntax Form:**  
[*line no.*] **RSMOVE** *expression*,*expression*

**Descriptive Form:**  
[*line no.*] **RSMOVE** *x displacement in device units*, *y displacement in device units*

**SDRAW** Draws a line from the current pointer position to a position specified in graphic device coordinates.

**Syntax Form:**  
[*line no.*] **SDRAW** *expression*,*expression*[,*expression*,*expression*] ...

**Descriptive Form:**  
[*line no.*] **SDRAW** *x coordinate in device units*, *y coordinate in device units*  
[,*x coordinate in device units*, *y coordinate in device units*] ...

**SEEVIE** Obtains the minimum and maximum x and y graphic device coordinates that define the current graphics viewport.

**Syntax Form:**  
[*line no.*] **SEEVIE** *variable*,*variable*,*variable*,*variable*

**Descriptive Form:**  
[*line no.*] **SEEVIE** *target for low x value*, *target for high x value*,  
*target for low y value*, *target for high y value*



**SEEWINDOW** Obtains the minimum and maximum x and y values that define the current graphics window.

**Syntax Form:**

[line no.] **SEEWIN** variable,variable,variable,variable

**Descriptive Form:**

[line no.] **SEEWINDOW** target for low x value, target for high x value,  
target for low y value, target for high y value

**SETGR**

Modifies default plots produced by GRAPH and XYPLOT commands.

**Syntax Form:**

[line no.] **SETGR** {  
    **GRAT** expression,expression[,expression,expression]  
    **NOGR**  
    **NOLA**  
    **NOPL**  
    **TICS** expression,expression[,expression,expression]  
    **VIEW**  
    **WIND**  
    **XOFF** expression  
}

[  
    ,  
    {  
        **GRAT** expression,expression[,expression,expression]  
        **NOGR**  
        **NOLA**  
        **NOPL**  
        **TICS** expression,expression[,expression,expression]  
        **VIEW**  
        **WIND**  
        **XOFF** expression  
    }  
    ...  
]

**Descriptive Form:**

[line no.] **SETGR**

**GRATICULE** major tic type for x axis,  
major tic type for y axis  
[,minor tic type for x axis,  
minor tic type for y axis]  
**NOGRATICULE**  
**NOLABEL**  
**NO PLOT**  
**TICS** number of major tic intervals for x axis,  
number of major tic intervals for y axis  
[,number of minor tic intervals for x axis,  
number of minor tic intervals for y axis]  
**VIEWPORT**  
**WINDOW**  
**XOFFSET** base value to add to x axis of graphics window

**GRATICULE** major tic type for x axis,  
major tic type for y axis  
[,minor tic type for x axis,  
minor tic type for y axis]  
**NOGRATICULE**  
**NOLABEL**  
**NO PLOT**  
**TICS** number of major tic intervals for x axis,  
number of major tic intervals for y axis  
[,number of minor tic intervals for x axis,  
number of minor tic intervals for y axis]  
**VIEWPORT**  
**WINDOW**  
**XOFFSET** base value to add to x axis of graphics window

...

**SGIN**

Inputs position of crosshair cursor in graphic device coordinates.

**Syntax Form:**

[line no.] **SGIN** string variable,variable,variable

**Descriptive Form:**

[line no.] **SGIN** target for input character,  
target for x coordinate of crosshairs in device units,  
target for y coordinate of crosshairs in device units

**SMOVE**

Moves pointer to position specified in graphic device coordinates.

**Syntax Form:**

[line no.] **SMOVE** expression,expression

**Descriptive Form:**

[line no.] **SMOVE** x coordinate in device units, y coordinate in device units

**VIEWPORT**

Specifies portion of graphics device space to be used for plotting data.

**Syntax Form:**

[line no.] **VIEWPO** expression,expression,expression,expression

**Descriptive Form:**

[line no.] **VIEWPORT** minimum x coordinate in device units,  
maximum x coordinate in device units,  
minimum y coordinate in device units,  
maximum y coordinate in device units



## WINDOW

Specifies range of user data to be drawn on selected viewport.

### Syntax Form:

[line no.] **WINDOW** expression, expression, expression, expression

### Descriptive Form:

[line no.] **WINDOW** minimum x coordinate in user units,  
maximum x coordinate in user units,  
minimum y coordinate in user units,  
maximum y coordinate in user units

## XYPLOT

Creates an X-Y plot of one array or waveform against a second one. Axes and labels accompany plot.

### Syntax Form:

[line no.] **XYPLOT**  $\left\{ \begin{array}{l} \text{array} \\ \text{waveform} \end{array} \right\}, \left\{ \begin{array}{l} \text{array} \\ \text{waveform} \end{array} \right\} \left[ \left\{ \begin{array}{l} \text{array} \\ \text{waveform} \end{array} \right\}, \left\{ \begin{array}{l} \text{array} \\ \text{waveform} \end{array} \right\} \right] \dots$

## R7912 DRIVER COMMANDS

### DEFECT

Acquires a composite of the target defects of an R7912.

#### Syntax Form:

[line no.] **DEFECT** #expression, simple numeric variable[, expression]

#### Descriptive Form:

[line no.] **DEFECT** #ilun, auto-dimensioned integer target for defect data  
[, number of samples]

### EDGE

Determines the edge-of-trace values for an R7912 waveform stored in standard format.

#### Syntax Form:

[line no.] **EDGE** integer array,  $\left\{ \begin{array}{l} \text{integer array} \\ \text{integer waveform} \end{array} \right\}$ ,  $\left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{integer array} \\ \text{integer waveform} \end{array} \right\}$ ,

$\left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{integer array} \\ \text{integer waveform} \end{array} \right\}$  [, expression, expression]

#### Descriptive Form:

[line no.] **EDGE** vertical value data, pointer table, target for upper-edge data,  
target for lower-edge data [, maximum trace width, rate of change]

## INSTALL

Uses dot graticule data to compute correction tables for geometry correction process.

### Syntax Form:

[line no.] **INSTALL** integer array,  $\left\{ \begin{array}{l} \text{integer array} \\ \text{integer waveform} \end{array} \right\}$ ,  $\left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{floating-point array} \end{array} \right\}$ ,  
 $\left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{floating-point array} \end{array} \right\}$

### Descriptive Form:

[line no.] **INSTALL** vertical value data, pointer table,  
target for horizontal correction table,  
target for vertical correction table

## MAP

Performs geometric correction of edge data from an R7912, using correction tables generated by **INSTALL** command.

### Syntax Form:

[line no.] **MAP**  $\left\{ \begin{array}{l} \text{array} \\ \text{waveform} \end{array} \right\}$ ,  $\left\{ \begin{array}{l} \text{array} \\ \text{waveform} \end{array} \right\}$ , floating-point array,  
floating-point array [,floating-point variable]

### Descriptive Form:

[line no.] **MAP** upper- or lower-edge data,  
target for geometry-corrected upper- or lower-edge data,  
horizontal correction table, vertical correction table  
[,target for maximum number of consecutive interpolated data values]



**NORMAL**

Converts edge data to a single-valued, zero-referenced, scaled waveform.

**Syntax Form:**

$$[\text{line no.}] \text{ NORMAL } \left\{ \begin{array}{c} \text{array} \\ \text{waveform} \end{array} \right\}, \left\{ \begin{array}{c} \text{array} \\ \text{waveform} \end{array} \right\}, \left\{ \begin{array}{c} \text{simple numeric variable} \\ \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\},$$

expression, expression [, floating-point variable]

**Descriptive Form:**

[line no.] **NORMAL** upper-edge data, lower-edge data, target for normalized data,  
zero-reference value, vertical scale factor  
[, target for number of consecutive interpolated data values]

**REJECT**

Flags data points in raw data which have been identified as defects by the DEFECT command.

**Syntax Form:**

$$[\text{line no.}] \text{ REJECT integer array, } \left\{ \begin{array}{c} \text{integer array} \\ \text{integer waveform} \end{array} \right\}, \left\{ \begin{array}{c} \text{simple numeric variable} \\ \text{integer array} \end{array} \right\}$$
**Descriptive Form:**

[line no.] **REJECT** vertical value data which is target for defect flags,  
pointer table, defect data

**TDPLOT**

Graphs raw R7912 data into current graphics window.

**Syntax Form:**

[line no.] **TDPLOT** integer array,  $\left\{ \begin{array}{l} \text{integer array} \\ \text{integer waveform} \end{array} \right\} [, \text{expression}]$

**Descriptive Form:**

[line no.] **TDPLOT** vertical value data, pointer table [,vertical scale factor]

**UNLOG**

Converts information data-logged from an R7912 to standard format.

**Syntax Form:**

[line no.] **UNLOG** integer array, string variable **TO** simple numeric variable,

$\left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{integer array} \\ \text{integer waveform} \end{array} \right\} , \text{floating-point variable}$

**Descriptive Form:**

[line no.] **UNLOG** raw data, knob readout **TO**  
auto-dimensioned integer target for vertical value data,  
target for pointer table, target for vertical scale factor

**ZREF**

Computes a zero-reference value from edge arrays.

**Syntax Form:**

[line no.] **ZREF**  $\left\{ \begin{array}{l} \text{array} \\ \text{waveform} \end{array} \right\} , \left\{ \begin{array}{l} \text{array} \\ \text{waveform} \end{array} \right\} , \text{floating-point variable}$

**Descriptive Form:**

[line no.] **ZREF** upper-edge data, lower-edge data, target for zero-reference value

## DPO ENVELOPE COMMAND

### ENVDP0

Acquires DPO arrays of signal minima and maxima.

#### Syntax Form:

[line no.] **ENVDP0** #expression,string expression[,expression]

#### Descriptive Form:

[line no.] **ENVDP0** #ilun, DPO source and target array specifications [,number of passes]



## 7912AD COMMANDS

### ADLOG

Acquires raw waveform data from a 7912AD in REPEAT mode and stores the data on a DEC RK05 hard disk.

#### Syntax Form:

[line no.] **ADLOG** #expression **FROM** @expression,expression,expression,expression[,**FAST**]

#### Descriptive Form:

[line no.] **ADLOG** #target plun **FROM** @IEEE 488 interface number, talk address,  
secondary address, number of times to digitize [,**FAST** mode switch]

### ADPLOT

Graphs raw 7912AD data into current graphics window.

#### Syntax Form:

[line no.] **ADPLOT** integer array,  $\left\{ \begin{array}{l} \text{integer array} \\ \text{integer waveform} \end{array} \right\} [,expression]$

#### Descriptive Form:

[line no.] **ADPLOT** vertical value data, pointer table [,vertical scale factor]

**EDGEAD**

Determines the edge-of-trace values for a 7912AD waveform stored in standard format.

**Syntax Form:**

$$[\text{line no.}] \text{EDGEAD integer array, } \left\{ \begin{array}{l} \text{integer array} \\ \text{integer waveform} \end{array} \right\}, \left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{integer array} \\ \text{integer waveform} \end{array} \right\},$$

$$\left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{integer array} \\ \text{integer waveform} \end{array} \right\} [,\text{expression},\text{expression}]$$
**Descriptive Form:**

[line no.] **EDGEAD** vertical value data, pointer table, target for upper-edge data,  
target for lower-edge data [,maximum trace width, rate of change]

**INSTAD**

Uses dot graticule data to compute correction tables for geometry correction process.

**Syntax Form:**

$$[\text{line no.}] \text{INSTAD integer array, } \left\{ \begin{array}{l} \text{integer array} \\ \text{integer waveform} \end{array} \right\},$$

$$\left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{floating-point array} \end{array} \right\}, \left\{ \begin{array}{l} \text{simple numeric variable} \\ \text{floating-point array} \end{array} \right\}$$
**Descriptive Form:**

[line no.] **INSTAD** vertical value data, pointer table,  
target for horizontal correction table,  
target for vertical correction table

**MAPAD**

Performs geometric correction of edge data from a 7912AD, using correction tables generated by the INSTAD command.

**Syntax Form:**

[line no.] **MAPAD**  $\left\{ \begin{array}{c} \text{array} \\ \text{waveform} \end{array} \right\}$ ,  $\left\{ \begin{array}{c} \text{array} \\ \text{waveform} \end{array} \right\}$ , floating-point array,  
floating-point array [,floating-point variable]

**Descriptive Form:**

[line no.] **MAPAD** upper- or lower-edge data, target for geometry-corrected upper- or lower-edge data, horizontal correction table, vertical correction table  
[,target for maximum number of consecutive interpolated data values]

**NORMAD**

Converts edge data to a single-valued, zero-referenced, scaled waveform.

**Syntax Form:**

[line no.] **NORMAD**  $\left\{ \begin{array}{c} \text{array} \\ \text{waveform} \end{array} \right\}$ ,  $\left\{ \begin{array}{c} \text{array} \\ \text{waveform} \end{array} \right\}$ ,  $\left\{ \begin{array}{c} \text{simple numeric variable} \\ \text{floating-point array} \\ \text{floating-point waveform} \end{array} \right\}$ ,  
expression,expression[,floating-point variable]

**Descriptive Form:**

[line no.] **NORMAD** upper-edge data, lower-edge data, target for normalized data, zero-reference value, vertical scale factor  
[,target for number of consecutive interpolated data values]



## REJECT

Flags data points in raw data which have been identified as defects by using the 7912AD READ DEF command.

### Syntax Form:

[line no.] **REJECT** integer array,  $\left\{ \begin{array}{l} \text{integer array} \\ \text{integer waveform} \end{array} \right\}, \text{ integer array}$

### Descriptive Form:

[line no.] **REJECT** vertical value data which is target for defect flags,  
pointer table, defect data

## ZREF

Computes a zero-reference value from edge arrays.

### Syntax Form:

[line no.] **ZREF**  $\left\{ \begin{array}{l} \text{array} \\ \text{waveform} \end{array} \right\}, \left\{ \begin{array}{l} \text{array} \\ \text{waveform} \end{array} \right\}, \text{floating-point variable}$

### Descriptive Form:

[line no.] **ZREF** upper-edge data, lower-edge data, target for zero-reference value

## HIGH-LEVEL SUPPORT COMMANDS

### BITCLR

Clears bits specified by second argument in the address specified by first argument.

#### Syntax Form:

$$[\text{line no.}] \text{ BITCLR } \left\{ \begin{array}{l} \text{expression} \\ \text{string expression} \end{array} \right\}, \left\{ \begin{array}{l} \text{expression} \\ \text{string expression} \end{array} \right\}$$

#### Descriptive Form:

$$[\text{line no.}] \text{ BITCLR } \left\{ \begin{array}{l} \text{decimal address} \\ \text{octal address} \end{array} \right\}, \left\{ \begin{array}{l} \text{decimal specification of bits to be cleared} \\ \text{octal specification of bits to be cleared} \end{array} \right\}$$

### BITSET

Sets bits specified by second argument in the address specified by first argument.

#### Syntax Form:

$$[\text{line no.}] \text{ BITSET } \left\{ \begin{array}{l} \text{expression} \\ \text{string expression} \end{array} \right\}, \left\{ \begin{array}{l} \text{expression} \\ \text{string expression} \end{array} \right\}$$

#### Descriptive Form:

$$[\text{line no.}] \text{ BITSET } \left\{ \begin{array}{l} \text{decimal address} \\ \text{octal address} \end{array} \right\}, \left\{ \begin{array}{l} \text{decimal specification of bits to be set} \\ \text{octal specification of bits to be set} \end{array} \right\}$$

**BITTST**

Determines if any bits specified by second argument are set in address specified by first. If so, sets third argument to 1; if not, sets it to 0.

**Syntax Form:**

$$[\text{line no.}] \text{ BITTST } \left\{ \begin{array}{l} \text{expression} \\ \text{string expression} \end{array} \right\}, \left\{ \begin{array}{l} \text{expression} \\ \text{string expression} \end{array} \right\}, \text{variable}$$
**Descriptive Form:**

$$[\text{line no.}] \text{ BITTST } \left\{ \begin{array}{l} \text{decimal address} \\ \text{octal address} \end{array} \right\}, \left\{ \begin{array}{l} \text{decimal specification of bits to be tested} \\ \text{octal specification of bits to be tested} \end{array} \right\},$$

target for test result

**HINPUT**

Inputs hexadecimal values to specified argument(s) from the terminal or other specified peripheral.

**Syntax Form:**

$$[\text{line no.}] \text{ HINPUT } [\# \text{expression},] \left\{ \begin{array}{l} \text{floating-point variable} \\ \text{floating-point array} \end{array} \right\} \left[ \left\{ \begin{array}{l} \text{floating-point variable} \\ \text{floating-point array} \end{array} \right\} \right] \dots$$
**Descriptive Form:**

$$[\text{line no.}] \text{ HINPUT } [\# \text{source plun},] \left\{ \begin{array}{l} \text{target floating-point variable} \\ \text{target floating-point array} \end{array} \right\}$$

$$\left[ \left\{ \begin{array}{l} \text{target floating-point variable} \\ \text{target floating-point array} \end{array} \right\} \right] \dots$$



## HPRINT

Prints argument(s) on terminal or other specified peripheral; any numeric arguments are printed in hexadecimal notation.

### Syntax Form:

[line no.] **HPRINT** [#expression,]  $\left[ \begin{array}{l} \text{string array} \\ \text{expression} \\ \text{array expression} \\ \text{waveform expression} \\ \text{string expression} \\ \text{TAB(expression)} \end{array} \right] \left\{ \begin{array}{l} , \\ ; \end{array} \right\} \left[ \begin{array}{l} \text{string array} \\ \text{expression} \\ \text{array expression} \\ \text{waveform expression} \\ \text{string expression} \\ \text{TAB(expression)} \end{array} \right] \dots$

### Descriptive Form:

[line no.] **HPRINT** [#target plun,]  $\left[ \begin{array}{l} \text{string array} \\ \text{expression} \\ \text{array expression} \\ \text{waveform expression} \\ \text{string expression} \\ \text{TAB(column number)} \end{array} \right] \left\{ \begin{array}{l} , \\ ; \end{array} \right\} \left[ \begin{array}{l} \text{string array} \\ \text{expression} \\ \text{array expression} \\ \text{waveform expression} \\ \text{string expression} \\ \text{TAB(column number)} \end{array} \right] \dots$

## HSET

Sets the variable to the hexadecimal value specified by the string expression.

### Syntax Form:

[line no.] **HSET** floating-point variable = string expression

### Descriptive Form:

[line no.] **HSET** target floating-point variable = hexadecimal value

## OINPUT

Inputs octal values to specified argument(s) from the terminal or other specified peripheral.

### Syntax Form:

$$[\text{line no.}] \text{ OINPUT } [\# \text{expression},] \left\{ \begin{array}{l} \text{floating-point variable} \\ \text{floating-point array} \end{array} \right\} \left[ \left\{ \begin{array}{l} \text{floating-point variable} \\ \text{floating-point array} \end{array} \right\} \right] \dots$$

### Descriptive Form:

$$[\text{line no.}] \text{ OINPUT } [\# \text{ source plun},] \left\{ \begin{array}{l} \text{target floating-point variable} \\ \text{target floating-point array} \end{array} \right\}$$
$$\left[ \left\{ \begin{array}{l} \text{target floating-point variable} \\ \text{target floating-point array} \end{array} \right\} \right] \dots$$

## OPRINT

Prints argument(s) on the terminal or other specified peripheral; any numeric arguments are printed in octal notation.

### Syntax Form:

[line no.] **OPRINT** [#expression,]  $\left[ \begin{array}{l} \text{string array} \\ \text{expression} \\ \text{array expression} \\ \text{waveform expression} \\ \text{string expression} \\ \text{TAB (expression)} \end{array} \right] \left\{ \begin{array}{l} , \\ ; \end{array} \right\} \left[ \begin{array}{l} \text{string array} \\ \text{expression} \\ \text{array expression} \\ \text{waveform expression} \\ \text{string expression} \\ \text{TAB (expression)} \end{array} \right] \dots$

### Descriptive Form:

[line no.] **OPRINT** [# target pln,]  $\left[ \begin{array}{l} \text{string array} \\ \text{expression} \\ \text{array expression} \\ \text{waveform expression} \\ \text{string expression} \\ \text{TAB (column number)} \end{array} \right] \left\{ \begin{array}{l} , \\ ; \end{array} \right\} \left[ \begin{array}{l} \text{string array} \\ \text{expression} \\ \text{array expression} \\ \text{waveform expression} \\ \text{string expression} \\ \text{TAB (column number)} \end{array} \right] \dots$

## OSET

Sets the variable to octal value specified by the string expression.

### Syntax Form:

[line no.] **OSET** floating-point variable = string expression

### Descriptive Form:

[line no.] **OSET** target floating-point variable = octal value



**RSTBUS** Performs a reset on the controller bus. All devices on the bus are set to their power-up state.

**Syntax Form:**

[line no.] **RSTBUS**

**VARCLR** Clears bits specified by second argument in the variable or array specified by first argument.

**Syntax Form:**

[line no.] **VARCLR**  $\left\{ \begin{array}{l} \text{variable} \\ \text{array} \end{array} \right\}, \left\{ \begin{array}{l} \text{expression} \\ \text{string expression} \end{array} \right\}$

**Descriptive Form:**

[line no.] **VARCLR**  $\left\{ \begin{array}{l} \text{target variable} \\ \text{target array} \end{array} \right\}, \left\{ \begin{array}{l} \text{decimal specification of bits to be cleared} \\ \text{octal specification of bits to be cleared} \end{array} \right\}$

**VARSET** Sets bits specified by second argument in the variable or array specified by first argument.

**Syntax Form:**

[line no.] **VARSET**  $\left\{ \begin{array}{l} \text{variable} \\ \text{array} \end{array} \right\}, \left\{ \begin{array}{l} \text{expression} \\ \text{string expression} \end{array} \right\}$

**Descriptive Form:**

[line no.] **VARSET**  $\left\{ \begin{array}{l} \text{target variable} \\ \text{target array} \end{array} \right\}, \left\{ \begin{array}{l} \text{decimal specification of bits to be set} \\ \text{octal specification of bits to be set} \end{array} \right\}$

## ASSEMBLY LEVEL SUPPORT COMMANDS

### GETR5

Returns the value of register 5 in specified variable.

#### Syntax Form:

[line no.] **GETR5** variable

#### Descriptive Form:

[line no.] **GETR5** target variable

### ODT

Enters the SPS ODT debugging program to permit debugging of resident or nonresident routines.

#### Syntax Form:

[line no.] **ODT**

### STAT

Prints the current status of the system plus the starting addresses of the nonresident commands and drivers in memory.

#### Syntax Form:

[line no.] **STAT**

[device name[constant]:[string expression][,SCHED]
string expression[,SCHED]
SCHED

#### Descriptive Form:

[line no.] **STAT**

[device name[drive number]:[target file name][,SCHED
target file name[,SCHED
SCHED

uler information flag]

## USEFUL FORMULAS

Note: E, X, Y, E1, E2, P1, P2, M, and N may be constants, variables, arrays, or expressions.

Function	TEK SPS BASIC Expression	Function	TEK SPS BASIC Expression
$\log_x$ of E	LOG(E)/LOG(X)	Arc cosine of E	ATN(SQR(1 - E * E)/E)
$\log_x^{-1}$ of E	X 1 E	Arc secant of E	ATN(SQR(E * E - 1))
Decibels		Arc cosecant of E	ATN(1/SQR(E * E - 1))
(voltage or current)	20 * LOG(E/E2)/LOG(10)	Arc cotangent of E	ATN(1/E)
(power)	10 * LOG(P1/P2)/LOG(10)	Hyperbolic sine of E	(EXP(E) - EXP(-E))/2
Rectangular to polar		Hyperbolic cosine of E	(EXP(E) + EXP(-E))/2
(X,Y to M/N)		Hyperbolic tangent of E	(EXP(E) - EXP(-E))/
(magnitude M)	SQR(X * X + Y * Y)		(EXP(E) + EXP(-E))
(angle N)	ATN(Y/X)	Hyperbolic secant of E	2/(EXP(E) + EXP(-E))
Polar to rectangular		Hyperbolic cosecant of E	2/(EXP(E) - EXP(-E))
(M/N to X,Y)		Hyperbolic cotangent of E	(EXP(E) + EXP(-E))/
(X coordinate)	M * COS(N)		(EXP(E) - EXP(-E))
(Y coordinate)	M * SIN(N)	Arc hyperbolic sine of E	LOG(E + SQR(E * E + 1))
Tangent of E	SIN(E)/COS(E)	Arc hyperbolic cosine of E	LOG(E + SQR(E * E - 1))
Secant of E	1/COS(E)	Arc hyperbolic tangent of E	(LOG(1 + E) - LOG(1 - E))/2
Cosecant of E	1/SIN(E)	Arc hyperbolic secant of E	LOG(1/E + SQR(1/E * E - 1))
Cotangent of E	COS(E)/SIN(E)	Arc hyperbolic cosecant of E	LOG(1/E + SQR(1/E * E + 1))
Arc sine of E	ATN(E/SQR(1 - E * E))	Arc hyperbolic cotangent of E	(LOG(E + 1) - LOG(E - 1))/2



## ABBREVIATED ERROR CODES

### PROGRAM CONTROL ERRORS

- C0** RUN command not in immediate mode.
- C1** Attempt to pass control to a nonexistent line number.
- C2** Attempt to overwrite program line being executed.
- C3** Program line exceeds 80 characters.
- C4** Priority value or task number is less than zero or greater than 126.
- C5** Concatenated statements are in an illegal order.
- C6** A line of program text with no line number was read from program file.
- C7** The line number of a subroutine scheduled with a SCHEDULE statement is not in memory.
- C8** Illegal ONERR condition when an error occurs.
- C9** ONERR RETURN statement encountered when no error has occurred.
- C10** An instrument or peripheral driver has been autoloading as if it were a nonresident command.
- C11** Illegal file contents.

### DATA ERRORS

- D0** Illegal data on input.
- D1** Number too large or too small.
- D2** String too long.
- D3** Source data types do not match destination specifications.
- D4** Simple string or numeric variable appears with a subscript, or string array is referenced without a subscript.
- D5** Arrays or waveforms of different lengths.
- D6** Subscript or zone boundary out of range.
- D7** Illegal waveform component.
- D8** Source waveforms do not have identical data sampling intervals or horizontal units.
- D9** Source items are not all waveforms; or if they are all waveforms, their data sampling interval and units are not identical.
- D10** Correction tables do not contain data required to perform geometry correction.
- D11** Illegal destination type.
- D12** Illegal source type.
- D13** Illegal address argument.
- D14** Array or waveform previously dimensioned to a different size.
- D15** Source data sampling interval (DSI) is too small.
- D16** Calculated data-record length is too long in a Record I/O form of a READU or WRITEU statement.



## ABBREVIATED ERROR CODES (cont.)

### EVALUATION ERRORS

- E0** Power operation performed on number less than or equal to zero.
- E1** Addition overflow.
- E2** Multiplication overflow.
- E3** Division overflow.
- E4** Floating-point-to-integer conversion overflow.
- E5** Double-to-single floating-point conversion overflow.
- E6** Addition underflow.
- E7** Multiplication underflow.
- E8** Division underflow.
- E9** Argument of EXP function is less than or equal to  $-89.0$ .
- E10** Divide by zero.
- E11** Argument of LOG function is less than or equal to zero.
- E12** Argument of EXP function is greater than 88.
- E13** Argument of SQR function is less than zero.
- E14** Underflow in power operation.
- E15** Overflow in power operation.
- E16** Arithmetic overflow during RFFT inverse transform setup.
- E17** Arithmetic underflow during RFFT inverse transform setup.
- E18** Divide by zero during RFFT inverse transform setup.
- E19** Arithmetic overflow during RFFT computation.
- E20** Arithmetic underflow during RFFT computation.
- E21** Divide by zero during RFFT computation.
- E22** Arithmetic overflow in recovery computations of CONV or CORR's direct transform.
- E23** Arithmetic underflow in recovery computations of CONV or CORR's direct transform.
- E24** Divide by zero in recovery computations of CONV or CORR's direct transform.
- E25** Arguments of ATAN2 command are both zero.

- E26** Arithmetic overflow during complex multiplication prior to inverse transform in CONV or CORR.
- E27** Arithmetic underflow during complex multiplication prior to inverse transform in CONV or CORR.
- E28** Divide by zero during complex multiplication prior to inverse transform in CONV or CORR.

### HARDWARE/SYSTEM ERRORS

- H0** Controller bus time-out.
- H1** Illegal controller instruction encountered.
- H2** Floating-point hardware malfunction.



## ABBREVIATED ERROR CODES (cont.)

### INSTRUMENT ERRORS

- I0** More than one type of device illegally sharing an interrupt vector.
- I1** Instrument driver is not in memory.
- I2** Illegal hardware unit number.
- I3** ILUN already attached to another instrument.
- I4** Instrument is not on line.
- I5** ILUN is not ATTACHed.
- I6** Illegal instrument function.
- I7** Write or timing error on output to a device on the IEEE 488 interface bus.
- I8** Interrupt specified in WHEN command occurred, but the specified line number is not in memory.
- I9** First horizontal address not found in data transfer from an R7912 Transient Digitizer.
- I10** Device and device driver are of different types.
- I11** Four DPOs already ATTACHed.
- I12** DPO bus time-out.
- I13** Device specified for R7912 Transient Digitizer fast data log not DK.
- I14** ILUN out of range.
- I15** The specified instrument is already ATTACHed to a different ILUN.
- I16** Reserved error code. It is not used in this version of BASIC.
- I17** Interrupt occurred on IEEE 488 interface bus for "ERR", "EOI", or "SRQ", but no interrupt condition exists.
- I18** Device on IEEE 488 interface bus did not accept or send data within the time-out period.
- I19** Insufficient data available for the variables specified.

### OPERATING SYSTEM ERRORS

- O0** Scheduler stack overflow.
- O1** Scheduler queue overflow.
- O2** Insufficient free memory.
- O3** Maximum number of nonresident commands, peripheral drivers, or instrument drivers has already been loaded.
- O4** String Functions deleted at load time.
- O5** Graphics option deleted at load time.
- O6** Auto-load feature not possible from system drive.
- O7** Temporary strings have been deleted while still in use.
- O8** Nonresident command or driver has attempted to move upper memory in order to obtain more room by releasing a buffer or deleting an array.
- O9** Clock queue overflow.
- O10** IEEE 488 (GPIB) capabilities deleted at system software load time.
- O11** Peripheral or instrument driver name is too long.
- O12** Nonresident module is incompatible with version of monitor.



## ABBREVIATED ERROR CODES (cont.)

### PERIPHERAL ERRORS

- P0** Illegal use of keyboard.
- P1** PLUN not OPEN FOR READ.
- P2** PLUN not OPEN FOR WRITE.
- P3** Logical end-of-file reached but no transfer of control provided via EOF command.
- P4** Command cannot execute because all available PLUNs are in use.
- P5** Referenced file already exists on medium.
- P6** Illegal use of the system device driver.
- P7** Peripheral driver referenced is not in memory.
- P8** Specified file is already OPEN.
- P9** Specified file does not exist.
- P10** Driver specified for RELEASE is still in use.
- P11** Physical end-of-file reached.
- P12** COPY command's source and destination files are the same, or one or both of the specified files are OPEN.
- P13** Illegal function for specified driver.
- P14** Illegal function with OPEN file.
- P15** Peripheral device not ready.
- P16** Device is full.
- P17** Device directory is full.
- P18** Hardware input/output error.
- P19** Illegal device number.
- P20** PLUN is out of range.
- P21** Physical bounds of flexible disk exceeded.
- P22** Unrecognized input/output media format.
- P23** Device not currently addressable.
- P24** PLUN specified in the record I/O form of a READU or WRITEU command is not OPEN FOR UPDATE.

### SYNTAX ERRORS

- S0** Illegal command name.
- S1** Illegal character in source statement.
- S2** Illegal item within parentheses or parentheses unmatched.
- S3** Operator or operand omitted in statement.
- S4** Illegal array zone use or illegal colon.
- S5** Illegal function argument.
- S6** Illegal driver specification.
- S7** No space after keyword.
- S8** Line number incorrectly used, omitted, or out of range.
- S9** Illegal numeric argument.
- S10** Illegal or missing delimiter.
- S11** Variable, array, subarray, waveform, or string variable not found where expected.
- S12** Missing keyword or keyword not found where expected.
- S13** Unmatched FOR/NEXT statements.
- S14** Illegal subscript or zone specification.
- S15** Illegal file name.
- S16** No equal sign where expected.
- S17** LET source type does not match destination type.
- S18** Illegal item in expression.
- S19** Illegal or missing relational operator in IF statement.
- S20** Operand does not match operator.
- S21** Illegal string function argument.
- S22** Illegal or missing command argument.
- S23** Too many parentheses in expression.



## BOOTING THE SYSTEM

### CP1100 Series Controllers with M9301 Bootstraps

Load Address 173000  
Press START

When dollar sign (\$) appears on terminal, type:

DKn—CP110 Disk Drive, n represents drive 0 through 7.  
DXn—CP112 or CP115 Dual Drive Floppy disk, n represents drive 0 or 1.  
Follow entry with a carriage return.

### CP1164 Series Controllers with M9301 Bootstraps

Press CNTRL-HALT  
Press CNTRL-BOOT

When dollar sign (\$) appears on terminal, type:

DKn—CP110 Disk Drive, n represents drive 0 through 7.  
DXn—CP112 or CP115 Dual Drive Floppy disk, n represents drive 0 or 1.  
Follow entry with a carriage return.

### CP4165 Controllers

Press RESTART  
When prompt appears, type:

DXn—CP112 or CP115 Dual Drive Floppy disk, n represents drive 0 or 1.

## CP4165 ODT COMMANDS

Command	Description
RETURN	Close opened location and accept next command.
LINE FEED	Close current location; open next sequential location.
↑	Open previous location.
←	Take contents of opened location, index by contents of PC, and open that location.
@	Take contents of opened location as absolute address and open that location.
r/	Open word at location r.
/	Reopen last location.
\$n/ or Rn/	Open general register n (0-7) or S (PS register).
r;G or rG	Go to location r and start program.
nL	Execute bootstrap loader using n as device command Status Register.
;P or P	Proceed with program execution.
RUBOUT	Erases previous numeric character. Response is a backslash (\).

## SYSTEM CONTROL CHARACTERS

Character	Action
Control-O	Alternately inhibits or allows the display of output directed to the system terminal.
Control-P	Terminates the program and returns BASIC to idle mode.
Control-U	Deletes line being entered at system terminal.
Control-Z	Terminates input to COPY command when the system terminal is the source device.

## ASCII & IEEE 488 (GPIB) CODE CHART

BITS B7 B6 B5 B4 B3 B2 B1				0 0 0 0				0 0 1 1				1 0 0 1				1 0 1 1				1 1 0 1				1 1 1 1			
				CONTROL				NUMBERS SYMBOLS				UPPER CASE				LOWER											
0 0 0 0	0	NUL	20	DLE	40	SP	60	0	100	@	120	P	140	\	160	p											
	0	(0)	10	(16)	20	(32)	30	(48)	40	(64)	50	(80)	60	(96)	70	(112)											
0 0 0 1	1	STL	21	LLO	41	!	61	1	101	A	121	Q	141	a	161	q											
	1	(1)	11	(17)	21	(33)	31	(49)	41	(65)	51	(81)	61	(97)	71	(113)											
0 0 1 0	2	STX	22	DC2	42	"	62	2	102	B	122	R	142	b	162	r											
	2	(2)	12	(18)	22	(34)	32	(50)	42	(66)	52	(82)	62	(98)	72	(114)											
0 0 1 1	3	ETX	23	DC3	43	#	63	3	103	C	123	S	143	c	163	s											
	3	(3)	13	(19)	23	(35)	33	(51)	43	(67)	53	(83)	63	(99)	73	(115)											
0 1 0 0	4	SDC	24	DCL	44	\$	64	4	104	D	124	T	144	d	164	t											
	4	(4)	14	(20)	24	(36)	34	(52)	44	(68)	54	(84)	64	(100)	74	(116)											
0 1 0 1	5	PPC	25	PPU	45	%	65	5	105	E	125	U	145	e	165	u											
	5	(5)	15	(21)	25	(37)	35	(53)	45	(69)	55	(85)	65	(101)	75	(117)											
0 1 1 0	6	ACK	26	SYN	46	&	66	6	106	F	126	V	146	f	166	v											
	6	(6)	16	(22)	26	(38)	36	(54)	46	(70)	56	(86)	66	(102)	76	(118)											
0 1 1 1	7	BEL	27	ETB	47	/	67	7	107	G	127	W	147	g	167	w											
	7	(7)	17	(23)	27	(39)	37	(55)	47	(71)	57	(87)	67	(103)	77	(119)											
1 0 0 0	10	GET	30	SPE	50	(	70	8	110	H	130	X	150	h	170	x											
	8	(8)	18	(24)	28	(40)	38	(56)	48	(72)	58	(88)	68	(104)	78	(120)											
1 0 0 1	11	TCT	31	SPD	51	)	71	9	111	I	131	Y	151	i	171	y											
	9	(9)	19	(25)	29	(41)	39	(57)	49	(73)	59	(89)	69	(105)	79	(121)											
1 0 1 0	12	LF	32	SUB	52	*	72	:	112	J	132	Z	152	j	172	z											
	10	(10)	1A	(26)	2A	(42)	3A	(58)	4A	(74)	5A	(90)	6A	(106)	7A	(122)											
1 0 1 1	13	VT	33	ESC	53	+	73	;	113	K	133	[	153	k	173	{											
	11	(11)	1B	(27)	2B	(43)	3B	(59)	4B	(75)	5B	(91)	6B	(107)	7B	(123)											
1 1 0 0	14	FF	34	FS	54	,	74	<	114	L	134	\	154	l	174	!											
	12	(12)	1C	(28)	2C	(44)	3C	(60)	4C	(76)	5C	(92)	6C	(108)	7C	(124)											
1 1 0 1	15	CR	35	GS	55	-	75	=	115	M	135	]	155	m	175	}											
	13	(13)	1D	(29)	2D	(45)	3D	(61)	4D	(77)	5D	(93)	6D	(109)	7D	(125)											
1 1 1 0	16	SO	36	RS	56	.	76	>	116	N	136	^	156	n	176	~											
	14	(14)	1E	(30)	2E	(46)	3E	(62)	4E	(78)	5E	(94)	6E	(110)	7E	(126)											
1 1 1 1	17	SI	37	US	57	/	77	? UNL	117	O	137	UNT	157	o	177	RUBOUT (DEL)											
	15	(15)	1F	(31)	2F	(47)	3F	(63)	4F	(79)	5F	(95)	6F	(111)	7F	(127)											



## INDEX

### A

ABORT command: 5  
ABS function: 3  
ADLOG command: 67  
ADPLOT command: 67  
ASC function: 3  
ASCII code chart: 84  
ATAN2 command: 5  
ATN function: 3  
ATTACH command: 5

### B

BITCLR command: 71  
BITSET command: 71  
BITTST command: 72  
BOOT command: 6  
Booting: 83

### C

CAN function: 3  
CANCEL command: 6  
CHAIN command: 6  
CHANGE command: 7  
CHR function: 3  
CLEAR command: 7  
CLOSE command: 7  
Control characters: 84  
Control errors: 79

CONVL command: 51  
COPY command: 8  
CORR command: 51  
COS function: 3  
CP4165: 83, 84  
CRS function: 3

### D

Data errors: 79  
DATE command: 8  
DEFECT command: 62  
DEFINE command: 9  
DELETE command: 10  
DETACH command: 11  
DIFF command: 52  
DIM command: 12  
DIR command: 13  
DISPLAY command: 54  
DRAW command: 54  
DRAWON command: 55  
Drivers: 2

### E

EDGE command: 61  
EDGEAD command: 67  
END command: 13  
ENVDPO command: 66  
EOF command: 13

Evaluation errors: 80  
EXP function: 3

### F

FOR command: 13  
FORMAT command: 14  
Formulas: 78  
Functions: 3

### G

GET command: 15  
GETBLK command: 16  
GETFREE command: 16  
GETLINE command: 16  
GETLOC command: 17  
GETPRIORITY command: 17  
GETR5 command: 77  
GETSTA command: 44  
GIFES command: 44  
GIN command: 55  
GOSUB command: 18  
GOTO command: 18  
GPIB code chart: 84  
GRAPH command: 55

### H

Hardware errors: 80  
HINPUT command: 72

## INDEX (Cont.)

HOOK command: 19  
HOOKQ command: 19  
HPRINT command: 73  
HSET command: 73

### I

IEEE 488 code chart: 84  
IF command: 20  
IFDTM command: 44  
IGNORE command: 20  
INITG command: 55  
INPREQ command: 21  
INPUT command: 21  
INSTAD command: 68  
INSTALL command: 63  
Instrument drivers: 2  
Instrument errors: 81  
INT command: 52  
INTEGER command: 22  
ITP function: 3

### J

### K

### L

LDA files: 2  
LEN function: 3  
LET command: 23  
LIST command: 23

LISTVAR command: 24  
LOAD command: 24  
LOCKKB command: 24  
LOG function: 3

### M

MAP command: 63  
MAPAD command: 69  
MATCH command: 24  
MAX function: 3  
MEA function: 3  
MIN function: 3  
MOVE command: 56

### N

NEXT command: 25  
NORMAD command: 69  
NORMAL command: 64

### O

ODT command: 77  
ODT commands for CP4165: 83  
OINPUT command: 74  
OLD command: 25  
ONERR command: 25  
OPEN command: 26  
Operating system errors: 81  
Operators: 3  
OPRINT command: 75

OSET command: 75  
OVERLAY command: 26  
OVLOAD command: 27  
OVLSAV command: 27

### P

PAGE command: 56  
Peripheral drivers: 2  
Peripheral errors: 82  
POLAR command: 52  
POLL command: 45  
POS function: 3  
PPOLL command: 45  
PRINT command: 28  
PRIORITY command: 28  
PUT command: 29  
PUTBLK command: 30  
PUTLOC command: 30

### Q

### R

RANDOM command: 31  
RASCII command: 46  
RBYTE command: 46  
RDRAW command: 56  
READ command: 31  
READU command: 32  
REJECT command: 64, 70

## INDEX (Cont.)

RELEASE command: 33  
REM command: 33  
RENAME command: 33  
RENUM command: 33  
REPLACE command: 34  
RESCHEDULE command: 34  
RESET command: 34  
RESETG command: 56  
RETURN command: 34  
REWIND command: 35  
RFFT command: 53  
RFFT2 command: 53  
RMOVE command: 56  
RMS function: 3  
RND function: 3  
RSDRAW command: 57  
RSMOVE command: 57  
RSTBUS command: 76  
RUN command: 35

### S

SAVE command: 35  
SCHEDULE command: 36  
SDRAW command: 57  
SEEVIEW command: 57  
SEEWINDOW command: 58  
SEG function: 3  
SETDATE command: 36  
SETGR command: 58  
SETTIME command: 36

SGIN command: 60  
SGN function: 3  
SIFCOM command: 46  
SIFLIN command: 46  
SIFTO command: 47  
SIN function: 3  
SIZ function: 3  
SMOVE command: 60  
SQR function: 3  
SQUISH command: 37  
STAT command: 77  
STATUS command: 37  
STERMC command: 47  
STOP command: 37  
STR function: 3  
Syntax errors: 82  
Syntax notation: 4  
SYSBLD command: 37

### T

TDPLOT command: 65  
TIFL command: 48  
TIME command: 38  
TRM function: 3  
TSK function: 3

### U

UNLOG command: 65  
UNSCHEDULE command: 38

### V

VAL function: 3  
VARCLR command: 76  
VARSET command: 76  
VARTST command: 39  
VERSION command: 39  
VIEWPORT command: 60

### W

WAIT command: 40  
WASCII command: 49  
WAVEFORM command: 40  
WBYTE command: 50  
WHEN command: 41  
WINDOW command: 61  
WRITE command: 41  
WRITEU command: 42

### X

XYPLOT command: 61

### Y

### Z

ZERO command: 43  
ZREF command: 65, 70



