DATAMAX UV-1
Zgrass GRAPHICS SYSTEM

Zgrass GLOSSARY

February 12, 1982

(C) Copyright 1982
Real Time Design, Inc.
531 Plymouth Court, Suite 102
Chicago, IL 60605
All Rights Reserved
COPYRIGHT NOTICE

(C) Copyright 1982 by Real Time Design, Inc. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the express written permission of Real Time Design, Inc., 531 Plymouth Court, Suite 102, Chicago, IL 60605 USA.

DISCLAIMER

Datamax, Inc. and Real Time Design, Inc. make no representations or warranties with respect to the contents hereof and specifically disclaim any implied warranties of merchantability or fitness for any particular purpose. Further, Datamax, Inc. and Real Time Design, Inc. reserve the right to revise this publication and to make changes from time to time in the content hereof without obligation of Datamax, Inc. and/or Real Time Design, Inc. to notify any person or organization of such revision or changes.

****

References are made throughout this documentation to the equipment listed below. We hereby acknowledge use of these names and/or trademarks in this publication.

-ADM 5 Dumb Terminal
  Video Display Unit
  Lear Siegler, Inc.
  Data Products Div.
  Anaheim, CA

-Micropolis Disk Drive
  Micropolis Corporation
  Canoga Park, CA

-Bit Pad One Data
  Tablet/Digitizer
  Summagraphics Corporation
  Fairfield, CT

-Mini-Winchester
  Disk Drive
  International Memories, Inc.
  Cupertino, CA

-Epson MX 80 Printer
  Epson America, Inc.
  Torrance, Ca

-CP/M (Control Program
  Monitor)
  Digital Research
  Pacific Grove, CA

-Datamax Electrohome
  RGB Monitor
  Datamax, Inc.
  Elk Grove Village, IL
**DATAMAX UV-1 Zgrass**
**RESIDENT Zgrass COMMANDS & FUNCTIONS**
February 12, 1982

<table>
<thead>
<tr>
<th><strong>GRAPHICS ARRAYS:</strong></th>
<th><strong>DISK:</strong></th>
<th><strong>INPUT/ OUTPUT:</strong></th>
<th><strong>PROGRAM FLOW:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
<td>DDELETE.BAK</td>
<td>RS232. AGET</td>
<td>.B</td>
</tr>
<tr>
<td>ARRAY.INT</td>
<td>DFETCH</td>
<td>RS232.APUT</td>
<td>.F</td>
</tr>
<tr>
<td>ARRAY.STR</td>
<td>DFETCH.ZAP</td>
<td>RS232.SGET</td>
<td>GOTO</td>
</tr>
<tr>
<td>BOX</td>
<td>DGET</td>
<td>RS232.END</td>
<td>IF</td>
</tr>
<tr>
<td>CENTER</td>
<td>DGET.BAK</td>
<td>RS232.SPUT</td>
<td>JUMP.ERR</td>
</tr>
<tr>
<td>CLEAR</td>
<td>DGET.OR</td>
<td>RS232.BGET</td>
<td>RETURN</td>
</tr>
<tr>
<td>CLEAR.CRT</td>
<td>DGET.XOR</td>
<td>RS232.BPUT</td>
<td>SKIP</td>
</tr>
<tr>
<td>CLEAR.WIND</td>
<td>DGET.FAST</td>
<td>RS232.RESET</td>
<td>STOP</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>DINIT</td>
<td>TERMINAL</td>
<td>TIMEOUT</td>
</tr>
<tr>
<td>DISPLAY.SCREEN</td>
<td>DLOAD</td>
<td></td>
<td>WAIT</td>
</tr>
<tr>
<td>DISPLAY.PAN</td>
<td>DLOAD.CLEAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LINE</td>
<td>DLOAD.SET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PATTERN</td>
<td>DLOAD.ZAP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PATTERN.FILL</td>
<td>DLOOK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MMOVE</td>
<td>DPUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MMOVE.UP</td>
<td>DPUT.TV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POINT</td>
<td>DSETUP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POINT.SNAP</td>
<td>DSETUP.RESET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POINT.PAN</td>
<td>DUSEMAP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCALE</td>
<td>DLOAD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCALE.SCR</td>
<td>DLOAD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCALE.PAN</td>
<td>DLOAD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCROLL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHRINK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNAP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRIPE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRIPE.STR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRIPE.OFF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEXT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WINDOW</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WINDOW.BOX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WINDOW.FULL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WINDOW.CENTER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBAKS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCREATE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(continued)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>MATH:</strong></th>
<th><strong>STRING MANIPULATION:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCCOS</td>
<td>ASCII</td>
</tr>
<tr>
<td>ARCSIN</td>
<td>LENGTH</td>
</tr>
<tr>
<td>ARCTAN</td>
<td>LPAD</td>
</tr>
<tr>
<td>COSINE</td>
<td>STRING</td>
</tr>
<tr>
<td>EXP</td>
<td>STRING.NUM</td>
</tr>
<tr>
<td>INT</td>
<td>SUBSTR</td>
</tr>
<tr>
<td>LENGTH.NUM</td>
<td></td>
</tr>
<tr>
<td>LN</td>
<td></td>
</tr>
<tr>
<td>LOG</td>
<td></td>
</tr>
<tr>
<td>POWER</td>
<td></td>
</tr>
<tr>
<td>SINE</td>
<td></td>
</tr>
<tr>
<td>SQR</td>
<td></td>
</tr>
<tr>
<td>TANGENT</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>USER INFO:</strong></th>
<th><strong>MISC.:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS</td>
<td>COMPARE</td>
</tr>
<tr>
<td>ADDRESS.AR</td>
<td>CONTROL</td>
</tr>
<tr>
<td>ADDRESS.STR</td>
<td>DELETE</td>
</tr>
<tr>
<td>ADDRESS.Z</td>
<td>DELETE.NULS</td>
</tr>
<tr>
<td>ANYARGS</td>
<td>EDIT</td>
</tr>
<tr>
<td>CORE</td>
<td>LOOPMAX</td>
</tr>
<tr>
<td>HELP</td>
<td>RESTART</td>
</tr>
<tr>
<td>STATUS</td>
<td></td>
</tr>
<tr>
<td>USEMAP</td>
<td></td>
</tr>
<tr>
<td>VERSION</td>
<td></td>
</tr>
</tbody>
</table>

(continued)
DATAMAX UV-1 Zgrass
SWAP COMMANDS and SWAP FUNCTIONS
February 12, 1982

<table>
<thead>
<tr>
<th>GRAPHICS/ARRAYS:</th>
<th>ERROR HANDLING:</th>
<th>STRING MANIPULATION:</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUILD</td>
<td>GETERROR</td>
<td>BUMP</td>
</tr>
<tr>
<td>ELLIPSE</td>
<td></td>
<td>MATCH</td>
</tr>
<tr>
<td>CMPARA</td>
<td></td>
<td>REPLACE</td>
</tr>
<tr>
<td>FONT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRAP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DISK:</th>
<th>MISCELLANEOUS:</th>
<th>USER INFO:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCHECK</td>
<td>ZAP1</td>
<td>DEBUG</td>
</tr>
<tr>
<td>DCOPY</td>
<td>ZAP2</td>
<td>SHOW</td>
</tr>
<tr>
<td>DDSMAP</td>
<td></td>
<td>WHATSIS</td>
</tr>
<tr>
<td>DFORMAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMATCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOWNER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRENAME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DZAP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## BUZZWORDS
(common computer terms)

<table>
<thead>
<tr>
<th>Term</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS</td>
<td>ITERATION</td>
</tr>
<tr>
<td>ALGORITHM</td>
<td>LOGICAL OPERATOR</td>
</tr>
<tr>
<td>AND</td>
<td>LOOP</td>
</tr>
<tr>
<td>ARGUMENT</td>
<td>MEMORY</td>
</tr>
<tr>
<td>ARGUMENT LIST</td>
<td>NUMBER</td>
</tr>
<tr>
<td>ASSIGNMENT</td>
<td>NUMERIC VARIABLE</td>
</tr>
<tr>
<td>ASSIGNMENT OPERATOR</td>
<td>OR</td>
</tr>
<tr>
<td>BIT</td>
<td>OVERFLOW</td>
</tr>
<tr>
<td>BYTE</td>
<td>PIXEL</td>
</tr>
<tr>
<td>BYTE ARRAY</td>
<td>PLOT</td>
</tr>
<tr>
<td>CALL</td>
<td>PRECEDENCE</td>
</tr>
<tr>
<td>CLIPPING</td>
<td>PRIORITY WRITE</td>
</tr>
<tr>
<td>COMMAND</td>
<td>PUNCTUATION</td>
</tr>
<tr>
<td>COMMENT</td>
<td>RADIANS</td>
</tr>
<tr>
<td>CONCATENATION</td>
<td>RANDOM</td>
</tr>
<tr>
<td>CONSTANT</td>
<td>RECURSION</td>
</tr>
<tr>
<td>CONTROL CHARACTERS</td>
<td>RELATIONAL OPERATOR</td>
</tr>
<tr>
<td>EXCLUSIVE OR</td>
<td>RESOLUTION</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>REVERSE PRIORITY</td>
</tr>
<tr>
<td>FILES</td>
<td>SEMANTICS</td>
</tr>
<tr>
<td>FLOATING POINT</td>
<td>SNAPPED PIX</td>
</tr>
<tr>
<td>FRAME BUFFER</td>
<td>STRING</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>STRING VARIABLE</td>
</tr>
<tr>
<td>INDEX</td>
<td>SWITCH</td>
</tr>
<tr>
<td>INDIirection</td>
<td>SYNTAX</td>
</tr>
<tr>
<td>INFINITE LOOP</td>
<td>TRUTH TABLES</td>
</tr>
<tr>
<td>INTEGER</td>
<td>VALUE</td>
</tr>
<tr>
<td>INTERRUPT</td>
<td>VARIABLE</td>
</tr>
<tr>
<td></td>
<td>WRAP AROUND</td>
</tr>
</tbody>
</table>

## IDIOSYNCRASIES
(special Zgrass terms)

<table>
<thead>
<tr>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABBREVIATION</td>
</tr>
<tr>
<td>CENTERING</td>
</tr>
<tr>
<td>(of graphics primitives)</td>
</tr>
<tr>
<td>COLOR</td>
</tr>
<tr>
<td>COLOR MAP</td>
</tr>
<tr>
<td>COLOR MODES</td>
</tr>
<tr>
<td>COORDINATES</td>
</tr>
<tr>
<td>CURSOR</td>
</tr>
<tr>
<td>DEVICE VARIABLES</td>
</tr>
<tr>
<td>DISK</td>
</tr>
<tr>
<td>DISPLAY MODES</td>
</tr>
<tr>
<td>ERROR NUMBER</td>
</tr>
<tr>
<td>EXPRESSION</td>
</tr>
<tr>
<td>JOYSTICK</td>
</tr>
<tr>
<td>LABEL</td>
</tr>
<tr>
<td>LOCAL VARIABLE</td>
</tr>
<tr>
<td>MACRO</td>
</tr>
<tr>
<td>NAME</td>
</tr>
<tr>
<td>NEXTLINE</td>
</tr>
<tr>
<td>OPERATOR</td>
</tr>
<tr>
<td>PANORAMA</td>
</tr>
<tr>
<td>PORTS</td>
</tr>
<tr>
<td>SCREEN</td>
</tr>
<tr>
<td>SWAP COMMAND</td>
</tr>
<tr>
<td>or FUNCTION</td>
</tr>
</tbody>
</table>

## SYSTEM MACROS

<table>
<thead>
<tr>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
</tr>
<tr>
<td>NC</td>
</tr>
<tr>
<td>ND</td>
</tr>
</tbody>
</table>
This page intentionally left blank
Note: BUZZWORDS are common computer terms. IDIOSYNCRASIES are concepts and features peculiar to or specially modified for ZGRASS. SWAP COMMANDs and SWAP FUNCTIONS have to be gotten from disk or tape first. SWITCHES modify commands. The ESOTERICA are the advanced features for experienced programmers.

ABBREVIATION
Idiosyncrasy

you can abbreviate COMMAND, FUNCTION, VARIABLE, and MACRO NAMES. For example:

```
PRINT 5
```

is the same as:

```
PR 5
```

This can cause confusion if you are not careful when you abbreviate NAMES.

Example:

```
TRY1=6
TR=2
```

will cause TRY1 to be equal to 2 because TR is a valid abbreviation for TRY1.

To verify this:

```
PRINT TR,TRY1
```

ABSOLUTE VALUE
Function

see "!" under OPERATOR.

ADDRESS
Esoteric Buzzword

the number which corresponds to the location of data in MEMORY.

ADDRESS(NAME)
Esoteric Function

returns an INTEGER which represents the ADDRESS of the NAME. Numbers in user memory are negative.

Example:

```
SAM=5
PR ADDRESS(SAM)
```

returns a number corresponding to SAM's address in decimal.
ADDRESS(NAME, NUMBER)
Esoteric Function
This allows you to read parts of the name header.

ADDRESS.STR(STRNAME, NUMBER)
Esoteric Function
returns the byte in the string corresponding to the number given. If a negative number is given, the string header can be read byte by byte.

ADDRESS.AR(ARNAME, NUMBER)
Esoteric Function
returns the byte in the array corresponding to the number given.

ADDRESS.Z(SWAPNAME, NUMBER)
Esoteric Function
returns the value of the byte in the swap module corresponding to the number given.

ADDRESS NAME, NUMBER, VALUE
Esoteric Command
puts the value (range 0-255) in the byte corresponding to the name plus the offset given by number.

ADDRESS.STR STRNAME, NUMBER, VALUE
Esoteric Command
puts the value (range 0-255) in the byte corresponding to the offset given by NUMBER past the name. This command is just like the STRING command.

ADDRESS.AR ARNAME, NUMBER, VALUE
Esoteric Command
puts the value (range 0-255) in the byte corresponding to the array name plus the offset given by NUMBER.

ADDRESS.Z SWAPNAME, NUMBER, VALUE
Esoteric Command
puts the value (range 0-255) in the byte corresponding to the swap module plus the offset given by NUMBER. You can use this one to painfully assemble a change to a swap module, if you know Z-80 assembler very well.
ALGORITHM
Buzzword
is a method you use to solve a problem.

AND
Buzzword
works on BITS. It makes 1's AND'ed with 1's equal to 1; and all other combinations produce 0.
AND table using 2 BITS:

<table>
<thead>
<tr>
<th></th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>===</td>
<td>====</td>
<td>====</td>
<td>====</td>
<td>====</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>

The AND COLOR MODES are 12-15. The AND DISPLAY MODES are 3,13,23,...,133,143.

ANYARGS()
Esoteric Function
returns 0 if no ARGUMENTs left in the ARGUMENT list passed to a MACRO and 1 if there are ARGUMENTs left in the ARGUMENT list. Note: ANYARGS is not compilable.
Example:
ADDEMUP=[SUM=O
IF ANYARGS()==1,INPUT A;SUM=SUM+A;SK 0
PRINT SUM]
ADDEMUP 5,10,15,20
ADDEMUP will add up all the arguments passed to it, and then print the total, which is 50 in this case.

ARCCOS(NUMBER)
Function
returns the inverse cosine of NUMBER.

ARCSIN(NUMBER)
Function
returns the inverse sine of NUMBER.
ARCTAN(NUMBER)
Function
returns the inverse tangent of NUMBER.

ARGUMENT
Buzzword
is computer talk for the stuff between commas that you give to a COMMAND, FUNCTION, or MACRO. (Actually, the first ARGUMENT has a space or ',' to its left and the last has a NEXTLINE, ';' or ') to its right, but there are always commas in between ARGUMENTs). ARGUMENTs must be VARIABLEs, NUMBERS, or EXPRESSIONs. Generally speaking, the presence of an ARGUMENT does not mean anyone is disagreeing about anything.
Note: superfluous spaces between ARGUMENTs and at the end of the line are not allowed. CTRL+Y will place a "!' at the end of each line marking the NEXTLINE so you can tell if there is an extra space between the last ARGUMENT and the NEXTLINE.

ARGUMENT LIST
Buzzword
is the list of ARGUMENTs that you give (pass) to a COMMAND, FUNCTION, or MACRO. You assign the passed ARGUMENTs to VARIABLEs in a MACRO by using the INPUT COMMAND (see INPUT).
Esoteric Note:
VARIABLEs are passed by NAME. Complex EXPRESSIONs (A+6-2) are EXECUTEd when they are passed. If you want to pass a VALUE, and the value is in a single VARIABLE (not an expression), use the "?" OPERATOR.
For instance:
A=10
PRINT A,A=100
will print 100,100. Since the ARGUMENTs are scanned before they get to PRINT.
A=10
PRINT ?A,A=100
will print 10,100
It is especially important to note that if LOCAL VARIABLEs are passed by NAME (no "?"), the called MACRO will not be able to access the LOCAL VARIABLE of the calling MACRO. If you must pass by VALUE, the following is an example of how to do it:
FEE=[a=100
FOO ?a]

FOO=[INPUT b
PRINT b*b]

Using "a+0" will also force evaluation for numerical VARIABLES. For STRINGS use "?" (for example, ?ABC), or CONCATENATE a null string. (i.e., ABC&[])

This problem shows up in global VARIABLES too. Compare:

TOM=[A=100
SAM A]

SAM=[A=10
INPUT B
PRINT B*B]

will print 100 whereas:

TOM=[A=100
SAM A+0]

will print 10000

If you want to force passing by VALUE, use the "?" OPERATOR. ZGRASS needs to be able to pass by NAME so the ASSIGNMENT OPERATOR can be used in EXPRESSIONs and so certain FUNCTIONS (like TABLET, for example) can return more than one VALUE.

ARRAY NAME,NUMBER
ARRAY NAME,N1,N2
ARRAY NAME,N1,N2,N3
ARRAY NAME,N1,N2,N3,N4

Command

creates a FLOATING POINT array with elements referenced by NAME(0), NAME(1),...,NAME(NUMBER-1). ARRAYS of one to four dimensions are specified by the one to four arguments given.

Example:

SHOW=[ARRAY JANE,200
A=0
JANE(A)=1%100
A=A+1
IF A<10,SK -2
CLEAR.C
USEMAP
A=0
PRINT "JANE("&A")="&JANE(A)
A=A+1
IF A<10,SKIP -2]
SHOW

When you run SHOW, it will first create the ARRAY JANE, then assign a RANDOM number to each element.
in JANE, then generate a USEMAP listing so you can see the size of JANE, and finally print out the first ten elements. If you change ARRAY JANE to ARRAY.INT JANE, you will notice USEMAP lists JANE as about half as big. For another ARRAY example see INDIRECTION. Note: LOCAL ARRAYS are allowed.

ARRAY.INT NAME, NUMBER
ARRAY.INT NAME, N1, N2
ARRAY.INT NAME, N1, N2, N3
ARRAY.INT NAME, N1, N2, N3, N4

Command
creates a FIXED POINT array with elements referenced by NAME(0), NAME(1),...,NAME(NUMBER-1). ARRAYS of one to four dimensions are specified by the one to four arguments given. Note: LOCAL ARRAYS are allowed.

Examples:
ARRAY.INT ROOTS, 10
will create a 10 element array referenced by ROOTS(0),...,ROOTS(9).

CARS={ARRAY.INT BUICK, 100
    A=0
    BUICK(A)=1%320
    A=A+1
    IF A<100, SK -2
    A=0
    BOX 0,0, BUICK(A), BUICK(A+1), 7
    A=A+2
    IF A<100, SK -2}
will fill an array, BUICK, with 100 RANDOM VALUES and use them to draw 50 BOXES.

ARRAY.INT CHECKER, 10, 10
will create a 100 element integer array referenced by CHECKER(0,0), CHECKER(0,1),...,CHECKER(9,9).
For another example, see INDIRECTION.

ARRAY.STR NAME, NUMBER
ARRAY.STR NAME, N1, N2
ARRAY.STR NAME, N1, N2, N3
ARRAY.STR NAME, N1, N2, N3, N4

Esoteric Command
creates a STRING array with string elements referenced by NAME(0), NAME(1),...,NAME(NUMBER-1). ARRAYS of one to four dimensions are specified by the one to four arguments given. To store STRING ARRAYS on tape or disk, you need to use GSTRING/PSTRING or GDSTRING/PDSTRING, SWAP MODULES which are not yet available.

Example:
This MACRO will fill the STRING ARRAY ATHRUZ with the letters A-Z and print them out. For another ARRAY example, see INDIRECTION. Note: LOCAL ARRAYS are allowed.

**ASCII(NUMBER)**

Esoteric Function returns a one character STRING corresponding to NUMBER, an ASCII value. ASCII is the coding system for characters, numbers and punctuation. Refer to a standard ASCII table for specific values. The STRING COMMAND takes characters and returns their ASCII values.

Example:

```plaintext
NUMS=[K:48
ZEROTONINE=ZEROTONINE&ASCII(K)
IF (K=K+1)<58,SK -1
PRINT ZEROTONINE]```

The ASCII values for the characters 0-9 are 48-57. This MACRO CONCATENATES the characters 0-9 and then prints them out as "0123456789".
ASCII VALUES FOR CONTROL CHARACTERS, NUMBERS, CAPITAL LETTERS, SMALL LETTERS, AND SYMBOLS

<table>
<thead>
<tr>
<th>ASCII Value</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>NUL</td>
</tr>
<tr>
<td>01</td>
<td>SOH</td>
</tr>
<tr>
<td>02</td>
<td>STX</td>
</tr>
<tr>
<td>03</td>
<td>ETX</td>
</tr>
<tr>
<td>04</td>
<td>EOT</td>
</tr>
<tr>
<td>05</td>
<td>ENQ</td>
</tr>
<tr>
<td>06</td>
<td>ACK</td>
</tr>
<tr>
<td>07</td>
<td>BEL</td>
</tr>
<tr>
<td>08</td>
<td>BS</td>
</tr>
<tr>
<td>09</td>
<td>HT</td>
</tr>
<tr>
<td>10</td>
<td>LF</td>
</tr>
<tr>
<td>11</td>
<td>VT</td>
</tr>
<tr>
<td>12</td>
<td>FF</td>
</tr>
<tr>
<td>13</td>
<td>CR</td>
</tr>
<tr>
<td>14</td>
<td>SO</td>
</tr>
<tr>
<td>15</td>
<td>SI</td>
</tr>
<tr>
<td>16</td>
<td>DLE</td>
</tr>
<tr>
<td>F17</td>
<td>DC1</td>
</tr>
<tr>
<td>F18</td>
<td>DC2</td>
</tr>
<tr>
<td>F319</td>
<td>DC3</td>
</tr>
<tr>
<td>F420</td>
<td>DC4</td>
</tr>
</tbody>
</table>

Note: RUB key has the ASCII value 127

The following macro prints the value of the key you press:

```plaintext
GIVEASCII=[
  IF (A=RS232(0))==0,SK 0
  PRINT A;SKIP -1]
```

ASSIGNMENT Buzzword

Examples:

- A=100
  This assigns the VALUE 100 to the VARIABLE A.
- LETTERS="ABCD"
  assigns the STRING "ABCD" to the VARIABLE LETTERS.
- LONGSTRING="THIS IS A VERY VERY LONG STRING WITH NEXTLINES AT THE END OF EVERY LINE. NOTICE YOU CAN HAVE NEXTLINES, COMMAS, PERIODS, AND ANY OTHER PUNCTUATION EXCEPT A DOUBLE QUOTE IN THIS CASE."
  Note that you can assign very long STRINGs to
VARIABLEs.

NULLSTRING=""

A VARIABLE can have a NULL STRING as its VALUE.

ROOT: (-B + SQRT(B*B*A*C))/2

EXPRESSIONs can be assigned to a VARIABLE.

You can put ASSIGNMENTS in EXPRESSIONs:

```
TOM: IF A<160, BOX 0,0, A=A+10, A, 3; SKIP 0]
```
system, sixteen BITS in an INTEGER and thirty two
bits in a floating point number.

BOX XCENTER, YCENTER, XSIZE, YSIZE, COLORMODE
Command
draws a filled rectangle of the dimensions XSIZE
by YSIZE, centered at XCENTER, YCENTER with drawing
mode specified by COLORMODE (see COLOR MODES for
the 21 options). If used as a function, a -1 is
returned if the bit is entirely off the screen;
and if an OR or XOR mode is used, a 0 is returned
if nothing non-zero was written over and a 1 is
returned if something was written over.
Example:
BOX 0, 0, 80, 60, 1
draws a rectangle centered at 0, 0 which is 80
PIXELS wide, 60 PIXELS high, and is drawn in
COLORMODE 1. If you draw a BOX which as a whole
can't fit on the screen, it will be CLIPPED to the
edges of the screen. For example:
BOX 150, 90, 100, 100, 1
will put a 60X60 BOX in the upper right corner.

BUILD NAME, XSCREEN, YSCREEN, COLORMODE, OVERRIDE
Esoteric Swap Command
creates a XSCREEN by YSCREEN panorama of screens
which are considered to be a "super snap." They
are stored in part or all of screens 4-15 under
the name given. The screens of the panorama will
be initialized as specified by the COLORMODE (0
means clear them, 1 means fill screen with $L1, 4
means xor them, that is, leave them as is, etc.).
The dimensions of the panorama are then taken as
XSCREEN*320, YSCREEN*201. The optional argument
OVERRIDE is specified if and only if all twelve
screens are to be reclaimed from a previously
built panorama.
Examples:
BUILD SAM, 3, 4, 0, 1
creates a "super snap" panorama 960 by 804 pixels
reclaiming space from any previous BUILDs.

BUILD TOM, 2, 3, 0
BUILD COPPER, 3, 2, 0
creates two "super snaps," the first 640 by 603
pixels, the second 720 by 402 pixels. Note: the
third argument is optional and taken as 0 if not
there. Of course, if you need to specify the
fourth argument (OVERRIDE), you have to specify
the third as well.
Use DISPLAY.PAN, PLACE, POINT.PAN, SCALE.PAN to access a "super snap."

Note: you cannot access "super snaps" as ARRAYS and you must explicitly delete a name created with BUILD before re-use. You may not use a name created with BUILD in any kind of assignment statement.
Also note that you can save individual screens of a "super snap" as screen dumps with DPUT.TV or you can save all of screens 4-15 on a diskette without any diskmap structuring by:

DLOAD.SET
DLOAD.ZAP

and then to retrieve it
DLOAD.CLEAR
DLOAD

followed by BUILDs with COLORMODE 4.

**BUMP STRING,NUMBER**

Esoteric Swap Command

increments the ASCII code of the last non-null character in a string by a specified numeric value.

Example:

```plaintext
TEST="ABCDE"
BUMP(TEST,2)
PRINT TEST
```

prints out the string "ABCDG"

Note: BUMP, unlike other string functions, does not cause a new copy of the string to be made.

Thus the following anomaly appears:

```plaintext
TEST="ABCDE"
BARB=TEST
BUMP(TEST,2)
PRINT TEST,BARB
```

will print "ABCDG" twice since BARB is still 'pointing at' the old but changed string. To make BARB be an entirely separate string, change the second line above to:

```plaintext
BARB=TEST[] this, if necessary: BARB=BARB[]
```

**BYTE**

Buzzword

a BYTE is the amount of MEMORY needed to hold a single character. Computers generally store one BYTE at each MEMORY location. ZGRASS lists the amount of MEMORY a NAMED thing takes up in BYTES when you use the USEMAP command.
BYTE ARRAY

**Buzzword**

if the values you want to store are limited to the range of 0-255 and you are very short on memory, you can use the STRING command as a way to store single byte values instead of characters. The STRING command can then be thought of as accessing the string as a BYTE ARRAY. If you place a zero in your BYTE ARRAY and attempt to store the string on the disk, it will only store as far as the zero. Be careful also not to print the string because some characters turn the terminal off, clear the screen, etc. This way of saving memory is for expert users only.

CALL

**Buzzword**

is what you do to cause the execution of a MACRO, COMMAND, or FUNCTION; that is, specifying its NAME and ARGUMENTS. ZGRASS has no CALL COMMAND since specifying a NAME plus ARGUMENTS is enough to call the MACRO, FUNCTION or COMMAND.

CENTERING (of Graphics)

**Idiosyncrasy**

The centering of even-numbered dimensions is biased to the upper right. The lower left hand corner of the upper right quadrant is the center pixel. For example, given a BOX centered at 0,0 which is 6 PIXELs wide on the X-axis, and 4 PIXELs high on the Y-axis, the left X would be -3, bottom Y -2, right X 2, top Y 1.

<--------6 PIXELs wide-------->

Y axis

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>-----------</td>
</tr>
</tbody>
</table>

-X

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
</tr>
<tr>
<td>-----------</td>
</tr>
</tbody>
</table>

-Y

You can see that the center PIXEL in this 6X4 box is located in the lower left hand corner of the upper right hand quadrant.
CIRCLE XCENTER,YCENTER,DIAMETER,0\1,COLORMODE
Command
draws a circle (specify 0 for border only, 1 for filled circle) centered at XCENTER, YCENTER with the specified DIAMETER using the COLORMODE indicated. Note: since the DIAMETER gives both width and height, you must use the ELLIPSE command to have unequal width and height. Also note that the CIRCLE command draws sort of flattened circles because the pixels are not quite square. Sorry.

CLEAR
Command
clears the TV screen (not the computer's memory). See FRAME BUFFER. RESTART clears the computer's memory, not the TV screen.

CLEAR.CRT
Command
clears the ADM-5 screen.

CLEAR.WIND
Command
clears the graphics WINDOW.

CLIPPING
Buzzword
refers to the action of displaying only a portion of a LINE, SNAP, BOX, CIRCLE, ELLIPSE, etc., if part of it exceeds the screen or window boundaries. Example:
BOX 120,80,100,100,3
will put a BOX in the upper right corner and throw away parts exceeding 159 in the X direction and 100 in the Y.

CMPARA(A1,A2)
Esoteric Swap Function
returns values depending on the comparison of two ARRAYs (usually used to compare SNAPs). The values returned are:
0 if all the BITs of A1<=A2
1 if all the BITs of A1==A2
-1 if all the BITs of A1>=A2
-2 otherwise
Example:
BOX 0,0,20,20,1
SNAP FIRST,0,0,20,20
BOX 0,0,20,20,3
SNAP SECOND,0,0,20,20
PRINT CMPARA(FIRST,SECOND)
prints 0 because all of FIRST is 01 PIXELs which are all less than or equal to all of SECOND's 11 PIXELs. If the second box were drawn in COLOR MODE 2, the result would be -2.

COLOR
Idiosyncrasy
The 256 COLORS available in ZGRASS form an abbreviated spectrum. You can get four COLORS on the screen at any one point. The default COLOR VALUES are white (7), red (91), green (165), and blue (18). By using the DEVICE VARIABLES $LO through $L3 you can change the currently available palette of 4 COLORS. The VALUE of $LO is 7 (white). The VALUE of $L1 is 91 (red), etc. See COLOR MAP for how ZGRASS keeps track of these four COLORS.

COLOR MAP
Idiosyncrasy
The COLOR MAP is the way ZGRASS translates COLORS 0-3 into the 256 available COLOR VALUES. The hardware looks at the values of $LO-$L3 before it writes a PIXEL to the screen. If it is writing a 0 it uses the COLOR VALUE (0-255) stored in $LO. If it is writing a 1, it uses the COLOR VALUE stored in $L1, and so on. To change the COLOR MAP so 1 refers to yellow instead of red, assign:

$L1=127

There are actually two COLOR MAPs, the $L's and the $R's. You get to the $R's by setting $HB. See DEVICE VARIABLES.

Example:
CBARS=[CLEAR; A=-149; C=0; $HB=21
$R0=0; $R1=82; $R2=43; $R3=249
$LO=7; $L1=213; $L2=126; $L3=164
IF A<115, BOX A=A+45,0,46,202,C=(C+1)%3+1; SK 0]

This will make a set of colorbars for tuning your TV.
COLOR MODES

Idiosyncrasy

The possible values for COLOR MODES are 0-21. You may need to study your truth tables for PLOP, XOR, OR, AND, PRIORITY, and REVERSE-PRIORITY logical operations to really understand what's going on. Look under PLOP, XOR, etc. for their respective truth table.

<table>
<thead>
<tr>
<th>COLOR MODE</th>
<th>MEANING:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PLOP with COLOR 00 (white)</td>
</tr>
<tr>
<td>1</td>
<td>PLOP with COLOR 01 (red)</td>
</tr>
<tr>
<td>2</td>
<td>PLOP with COLOR 10 (green)</td>
</tr>
<tr>
<td>3</td>
<td>PLOP with COLOR 11 (blue)</td>
</tr>
<tr>
<td>4</td>
<td>XOR screen with COLOR 0 (no change)</td>
</tr>
<tr>
<td>5</td>
<td>XOR screen with COLOR 1</td>
</tr>
<tr>
<td>6</td>
<td>XOR screen with COLOR 2</td>
</tr>
<tr>
<td>7</td>
<td>XOR screen with COLOR 3</td>
</tr>
<tr>
<td>8</td>
<td>OR with 00 (no change)</td>
</tr>
<tr>
<td>9</td>
<td>OR with 01 (if white or red, turn red if green or blue, turn blue)</td>
</tr>
<tr>
<td>10</td>
<td>OR with 10 (if white or green, turn green, if red or blue, turn blue)</td>
</tr>
<tr>
<td>11</td>
<td>OR with 11 (turn blue)</td>
</tr>
<tr>
<td>12</td>
<td>AND with 00 (turn white)</td>
</tr>
<tr>
<td>13</td>
<td>AND with 01 (if white or green turn white, if red or blue, turn red)</td>
</tr>
<tr>
<td>14</td>
<td>AND with 10 (if white or red, turn white, if green or blue, turn green)</td>
</tr>
<tr>
<td>15</td>
<td>AND with 11 (no change)</td>
</tr>
<tr>
<td>16</td>
<td>PRIORITY WRITE 01 (if white or red turn red, if green stay green, if blue stay blue)</td>
</tr>
<tr>
<td>17</td>
<td>PRIORITY WRITE 10 (if white, red or green turn green, if blue stay blue)</td>
</tr>
<tr>
<td>18</td>
<td>REVERSE-PRIORITY 01 (red, green, and blue turn red, and white stays white)</td>
</tr>
<tr>
<td>19</td>
<td>REVERSE-PRIORITY 10 (green and blue, turn green, red stays red, and white stays white)</td>
</tr>
<tr>
<td>20</td>
<td>Increment COLOR (if white turn red, if red turn green, if green turn blue, if blue turn white)</td>
</tr>
<tr>
<td>21</td>
<td>Decrement COLOR (if white turn blue, if red turn white, if green turn red, if blue turn green)</td>
</tr>
</tbody>
</table>
COMMAND

Buzzword

there are three types of COMMANDs: system COMMANDs, SWAP COMMANDs, and ones you define yourself, called MACROs. System COMMANDs are built-in and are listed by the HELP COMMAND. Swap COMMANDs function like System COMMANDs except they must first be gotten from tape or disk.

COMMENT

Buzzword

it is helpful to have COMMENTS in your MACROs to tell how they work. In ZGRASS, a line which starts with a '.' is taken as a COMMENT. You can also have COMMENTS on lines where there are COMMANDs by using a '; ' and then a '. '. Examples:

.THIS LINE IS TAKEN AS A COMMENT
LINE 6,-70,1; THIS LINE HAS A COMMAND TOO

COMPILE NAME, NEWNAME

Command

takes a MACRO called NAME, and creates a compiled MACRO called NEWNAME. Compiled MACROs are larger but run faster. They cannot be stored on disk or tape.

Note: several COMMANDs; EDIT, CORE, HELP, LOOPMAX, ONERROR, ANYARGS, WAIT, and USEMAP if included in a MACRO will cause your MACRO not to COMPILE and you will get ERROR #59.

Example:

TALL=[ARRAY LONGNAME,200
INDEX=0
LONGNAME(INDEX)=SQRT(INDEX)
INDEX=INDEX+1
IF INDEX<200,SKIP -2]

TALL will take approximately 19.1 seconds to run.

COMPILE TALL, FASTER

FASTER will take approximately 3.8 seconds to run. The compiler figures out NAME references, SKIPS, GOTOs, and figures out OPERATORs and parentheses. You will see better improvements in compiling when you have long programs with lots of arithmetic and/or long NAMES, or lots of LOCAL VARIABLES. COMPILEling BOX COMMANDs, on the other hand, gives a less dramatic speed increase because the time is spent mostly writing to the screen, not figuring out the ARGUMENTs. You can't store COMPILED MACROs on disk or tape.

Note that you should not compile macros from within compiled macros. Never never delete anything referenced in a compiled macro if you
expect to use it again unless you re-compile.

COMPRESS FONTARRAY, NAME

Swap Command

compresses the snaps in a FONTARRAY and creates a new FONTARRAY called NAME. COMPRESS allows single-color characters to be displayed with text in any color and also halves the space required. Any character in the font with more than one color will not be COMPRESSed. When a new character is added to an already COMPRESSed array, simply COMPRESS the array again and it will COMPRESS the new character as you might hope. Multiple COMPRESSes do not confuse the array.

CONCATENATION

Buzzword is joining STRINGs together with the '&' operator.

Examples:
- PRINT "A"&"B"&"C" prints ABC
- PRINT "A"&10 prints A10
- N="MOON"
- S="SHINE"
- PRINT N&S prints MOONSHINE

CONSTANT

Buzzword Examples:
- PRINT 'THIS is a constant or literal STRING'
- PRINT 33.75
- PRINT 1.23E17

Constants, unlike VARIABLEs, never change. You can have both NUMBERS and STRINGS as constants.

CONTROL CHARACTERS

Buzzword are single character requests you type on the keyboard by holding the key marked CTRL down (as you would the shift key) and at the same time pushing any key from A to Z. The CONTROL function and CONTROL command are used for programmatically reading and writing these characters. See the list of all the CONTROL characters on the next page(s).
**CONTROL(NUMBER)**

Esoteric Function
returns the current value of the CONTROL CHARACTER identified by NUMBER. For instance, to see if CTRL+Y is on:

```plaintext
PRINT CONTROL(25)
```

if CONTROL+Y is on, the answer will be 1, and if it is off, 0.

<table>
<thead>
<tr>
<th>CONTROL</th>
<th>CHAR.</th>
<th>NUM.</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>S</td>
<td>;Editor delete line;</td>
<td>also execute last line repetitively</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(mapped into TAB key on ADM-5)</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>*</td>
<td>;Resets COLORS to WRGB and $TV,$MW, $MR, and $ML to 0 and $HB to 44</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>S</td>
<td>;Stops currently running MACRO(s) and clears CONTROL characters</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>T</td>
<td>;Single step in MACROs on/off with CTRL+X gives single step and listing</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>5</td>
<td>S</td>
<td>;Editor exit and update and stops</td>
<td>PATTERN and PATTERN.FILL gracefully</td>
</tr>
<tr>
<td>F</td>
<td>6</td>
<td>S</td>
<td>;Editor copy lines</td>
<td>also re-edit last line typed</td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td>*</td>
<td>;Set all CTRL characters to 0</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>11</td>
<td>S</td>
<td>;Cursor Control</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>1</td>
<td>S</td>
<td>;Repeats the last command line once</td>
<td>same as CTRL+A and TAB key</td>
</tr>
<tr>
<td>J</td>
<td>9</td>
<td>S</td>
<td>;Cursor Control Cursor Control</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>10</td>
<td>S</td>
<td>;Cursor Control</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>8</td>
<td>L</td>
<td>;Editor Cursor Control</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>13</td>
<td>S</td>
<td>;Carriage return</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>14</td>
<td>T</td>
<td>;Beep on/off for CR</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>15</td>
<td>T</td>
<td>;Supress/allow printing on CRT</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>16</td>
<td>T</td>
<td>;Echo CRT on printer, if any</td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>17</td>
<td>T</td>
<td>;Start/Halt printing on CRT</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>18</td>
<td>S</td>
<td>;Delete character</td>
<td>also is ESC key on ADM-5</td>
</tr>
<tr>
<td>S</td>
<td>19</td>
<td>S</td>
<td>;Editor set move pointers</td>
<td>also execute last line once again</td>
</tr>
<tr>
<td>T</td>
<td>20</td>
<td>S</td>
<td>;Editor delete move pointers</td>
<td></td>
</tr>
<tr>
<td>U</td>
<td>21</td>
<td>*</td>
<td>;Line erase (outside the editor)</td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>22</td>
<td>T</td>
<td>;Allows auxiliary RS232 input</td>
<td>in parallel with keyboard RS232 input</td>
</tr>
<tr>
<td>W</td>
<td>23</td>
<td>T</td>
<td>;Twenty line mode on/off</td>
<td>waits for return key to print 20 more lines</td>
</tr>
<tr>
<td>X</td>
<td>24</td>
<td>T</td>
<td>;List on/off as MACRO EXECUTES</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>25</td>
<td>T</td>
<td>;A &quot;!&quot; is put at the end of every</td>
<td></td>
</tr>
</tbody>
</table>
;line which has a CR (in editor, ;also use CTRL+T)
Z 26 S ;Stop MACRO in progress and ;accept lines till return key alone ;typed
[ 27 S ;ESC key--same as CTRL+R
\ 28 T ;Switch upper/lower case
] 29 T ;Cancel/enable break button on ADM-5
^ 30 ;Not generated by ADM-5
RUB 31 T ;Allows .B macros to be interleaved ;with regular macros

TYPES:
T is a toggle switch which you can turn on (1) or off (0) by pressing the appropriate key an odd or even number of times. Use the CONTROL command to set these programmatically.
S can be set to (1) but not set to 0 by keyboard action. Use the CONTROL command to set these programmatically.
* means this CONTROL CHARACTER action is not accessible through the CONTROL COMMAND.
Note: the editing CONTROL Characters can be reset by using the TERMINAL Command.

CONTROL NUMBER1,NUMBER2
Esoteric Command
Like CONTROL (NUMBER) but it writes NUMBER2 in the CONTROL CHARACTER indicated by NUMBER1. Use to set CONTROL CHARACTERS in a MACRO. (Setting CONTROL CHARACTERS B,G,U to 1 doesn't do anything, however.) CONTROL CHARACTERS used only in EDIT (J,L,T) may be used by you for your own purposes outside of EDIT.

Characters D,N,O,P,Q,V,W,X,Y and [,,\],RUB are set to one by an odd number of user CTRL key presses and cleared to zero by even presses. The rest are set by one or more user presses and cleared by system actions.
Examples:
CONTROL 3,1; Will cause a CTRL+C to happen programmatically
CONTROL 16,1; Will cause whatever comes out on the CRT to be printed on the printer, if any.
CONTROL 15,1; Will cause whatever you type on the computer terminal to be not printed to the CRT until CONTROL 15,0 is EXECUTEd.
CONTROL 24,1; Will cause listing of lines as they EXECUTE until CONTROL 24,0 is EXECUTEd.
COORDINATES

Idiosyncrasy

are the values across the X (horizontal) axis and up and down the Y (vertical) axis. The COORDINATES range from -32768 to 32767. With the default WINDOW in effect the visible X-COORDINATES range from -160 to 159, and the Y-COORDINATES range from -100 to 100. See WINDOW.

CORE

Command

tells you how much memory you have in BYTES in how many fragments. The first number is the hexadecimal ADDRESS which you should ignore. A BYTE will hold one character so if you have a MACRO on tape or disk that is 500 BYTES long (USEMAP will give its length once it's in memory), CORE has to show a fragment with at least 500 BYTES for you to GETTAPE or DGET the MACRO without getting ERROR #27 (not enough memory space).

CORE()

Function

returns the size of the largest block of MEMORY left and also prints the CORE map. (You can suppress the printing with CONTROL 15,1.)

Example:

A=CORE()

will print a list of the available memory

PRINT A

will print 4064 if this is done right after RESTART.

COSINE(NUMBER)

Function

returns the cosine of NUMBER.

CURSOR

Idiosyncrasy

is the little white box on the ADM-5 indicating where the action of the next key press will take place. Both typing and edit functions move the CURSOR around.
DEBUG
Esoteric Swap Command
Refer to the Swap Module creation documentation, a separate package.

DELETE NAMEO, NAME1, NAME2, ..., NAMEn
Command
deletes the NAME/s (VARIABLE, ARRAY, STRING) from memory and reclaims the memory for further use.
Certain things cannot be deleted (DEVICE VARIABLES, the VARIABLES A-Z, system COMMANDS, and FUNCTIONS) so an appropriate ERROR message accompanies illegal deletion requests. Never DELETE anything that is referenced in a COMPILED MACRO unless you have already DELETED that COMPILED MACRO or intend not to use it again.
Example:
GONE="WITH THE WIND"
USEMAP will tell you that there is a STRING called GONE in MEMORY.
DELETE GONE
USEMAP will now show you that GONE is gone.

DELETE NULLS
Esoteric Command Switch
will get rid of all the null names hanging around. This is normally done while the system is waiting for you to type at normal command level but if you are running a very long macro that calls in dozens or hundreds of names, you may want to get rid of them periodically since they take up space. Don't use this feature unless you need to.

DEVICE VARIABLES
Idiosyncrasy
are special VARIABLES starting with a '$' that access system features. You use them just like other VARIABLES. Most DEVICE VARIABLES (except COLOR VARIABLES) are set to 0 when the system is turned on or reset.
VARIABLE: Description: Range:

Screen COLOR VARIABLES:

$LO COLOR 0 left 0-255
$L1 COLOR 1 left 0-255
$L2 COLOR 2 left 0-255
$L3 COLOR 3 left 0-255
(left means left half of screen set by $HB)
$R0$ COLOR 0 right 0-255
$R1$ COLOR 1 right 0-255
$R2$ COLOR 2 right 0-255
$R3$ COLOR 3 right 0-255

('right' means right half of screen, set by $HB$)

$HB$ Horizontal Color 0-44

$BC$ Border Color 0-3
0 set Border to $L0$
1 set Border to $L1$
2 set Border to $L2$
3 set Border to $L3$

JOYSTICK control VARIABLES:

$X1$-$X4$ X of JOYSTICKs 1-4 -1,0,1
$Y1$-$Y4$ Y of JOYSTICKs 1-4 -1,0,1
$K1$-$K4$ knob value of JOYSTICKs 1-4
$T1$-$T4$ trigger value of JOYSTICKs 1-4

DISK information:

$DS$ has disk number set by DSETUP -3 to 7
$DV$ disk verify on write: 0 = on

System Timers:

NOTE: system timers are suspended by tape I/O and floppy disk I/O operations

$Z0$-$Z9$ decremented by 1 every 1/60 second until 0

$TK$ system time in 1/60's seconds up to 60
$SC$ in seconds up to 60
$MN$ in minutes up to 60
$HR$ in hours up to 24
$DA$ in days up to 32767
$ST$ in seconds up to 32767

Example:

CLOCK=[PR $HR,'$:',$MN,'':',$SC,'':',$TK;SK 0]
CLOCK.B

Terminal Control:

$RS$ if non-zero, allows the high-order bit through from the RS232 ports; if zero, the high-order
bit is always 0

256K Screen Memory Controls (for an example, see SCREEN):
- $TV$ sets the screen the TV uses to 0-15
- $MW$ sets the screen the computer writes to
  and if $ML=0$, reads from 0-15
- $MR$ sets the screen the computer reads from
  if $ML=1$
- $ML$ if 1, allows read and write to
  be from different screens; if 0,
  forces $MW$ to be used for both read
  and write

Math Control:
- $RD$ if 0, use degrees;
  if 1, use radians

Graphic Control:
- $DX$ is the X offset for all graphic commands
- $DY$ is the Y offset for all graphic commands

Memory Allocation:
- $BF$ if non-zero, attempt to do a best-fit
  allocation, which takes longer but
  reduces memory fragmentation

Number Formatting:
- $KZ$ if 1, allows trailing zeroes after
  decimal point

Error Reporting:
- $ER$ stores the last ERROR NUMBER generated

DISK
Idiosyncrasy

A DISK (also called FLOPPY DISK or WINCHESTER DISK) is the best place to store information. Since it is a much more complex device than an audio tape recorder, several commands are necessary to manage it. You must occasionally do housekeeping on your disk to keep it from filling up. (Esoteric note: the disk software uses 512-byte sectors.) The umpteen disk commands are grouped as follows:

Resident Commands:
- to choose a disk, use DSETUP
- to reset the disk, use DSETUP.RESET
- to get a disk file, use DGET
to put a disk file, use DPUT
  to put out a screen dump, use DPUT.TV
  to delete a disk file, use DDELETE
  to initialize a disk, use DINIT
  to tell what is on the disk, use DUSEMAPI
  to create a submap name, use DCREATE
  to get into a submap, use DSETUP
  to load a whole disk into screens 4-15, use DLOAD
  to unload (write) a whole floppy, use DLOAD.ZAP
  to clear the floppy in memory without writing
  it, use DLOAD.CLEAR
  to make screens 4-15 think they have been
  DLOAD'd, use DLOAD.SET
  to lookup a file number, use DLOCK
  to get a file given its number, use DGET.FAST
  to check that the disk is readable, use DFETCH
  to load a specific sector, use DFETCH.NUMBER
  to write a specific sector, use DFETCH.ZAP
     NUMBER
  to delete all the BAKS, use DBAKS

Swap Commands:

to check a disk, use DCHECK
  to copy a disk to another disk, use DCOPY
  to rename a file name, use DRENAME
  to delete a whole submap, use DDSMAP
  to match file names, use DMATCH
  to read/write individual bytes, use DZAP
  to find out which file a sector belongs to, use
     DOWNER
  to format a disk, use DFORMAT

DBAKS Command

deletes all BAK files on the disk. DPUT
  automatically creates BAK files for you and these
  take up space. You can individually delete them
  with DDELETE.BAK or delete them all at once with
  DBAKS.

DCHECK Swap Command

reclaims any 'lost' sectors on the disk. Sectors
  can get lost if you push the red RST button during
  a DPUT or get an error during a DPUT. DCHECK does
  not verify the integrity of the data on the disk.
  See DCOPY and DFETCH.
DCOPY SOURCEDISK, NEWDISK

Swap Command

copies the SOURCEDISK onto the NEWDISK clobbering all previous information on the NEWDISK. The NEWDISK does not have to be DINIT'd but it must have been DFORMAT'd. DCOPY also verifies the information on the SOURCEDISK and NEWDISK (if $DV=O) as it is copying. You should backup disks with DCOPY fairly often (every couple hours of working) since floppies are not super-reliable. You can see the disk sectors numbers in the display lights.

Example:

DCOPY 0,1

copies what is on disk 0 to disk 1.

Copies can be made from or to DLOAD'd disks.

DCREATE SUBMAPNAME, [MESSAGE]

Command

creates a submapname on the disk. Submaps allow you to have several independent groupings of disk files on the same disk, thus allowing the same name to be in different submaps. Once you DCREATE a submapname, you will see it in DUSEMBAP. You then use DSETUP with a disk number and a submapname to make all disk commands reference only files within that submap (the exception is that if the command cannot find a name it looks in the normal (unnamed) map so it is easy to get swaps and common macros). If you DSETUP for a particular submapname without having DCREATE'd it, you may not be able to find DPUT'd files unless you are very good at remembering since the submapname will not show up in the normal disk map. DCREATE automatically puts you in the submapname you specified. Also, DCREATE does not check if there is already a submapname identical to the one you specify, so it is possible but not harmful to have two or more submaps with the same name.

Examples:

DSETUP 1 ;.setup for disk 1
DCREATE JOB77 ;.create submap JOB77
DDELETE FILENAME
DDELETE.BAK FILENAME
Command

deltes the FILENAME from the disk. If there is a
BAK file, it is not deleted. To delete a BAK
file, use DDELETE.BAK FILENAME. NOTE: If the
FILENAME is a name in memory, abbreviations will
work; nevertheless, you should not use
abbreviations for FILENAMEs.

DDSMAp SUBMAPNAME
Swap Command

deletes the submapname and all the files in the
submap. Be careful!

DFETCH
DFETCH NUMBER
DFETCH.ZAP NUMBER
Command

DFETCH by itself reads all 384 sectors on the
floppy disk; if all read ok, it says "OK";
otherwise, an error is printed, giving the sector
it can't read. If it errors out, the disk or disk
drive is bad. DFETCH NUMBER reads a specific
sector into the system memory reserved for the
disk. DFETCH.ZAP NUMBER writes the contents of
the system memory reserved for the disk onto a
disk. Use with care. These commands can be used
to copy part of a damaged disk. DOWNER can be
used to tell which filename a bad sector belongs
to so you know which file cannot be saved by a
DFETCH/DFETCH.ZAP copy loop.

DFORMAT DRIVENUMBER
Swap Command

formats the disk specified by NUMBER. Formatting
is the first step when using a brand-new disk or
reclaiming one that has lots of errors. Formatting erases all information on the disk. Use DINIT or DCOPY to make the disks
Zgrass-compatible after DFORMAT'ing them.

To format a brand-new disk, do the following:

1. Put your swap diskette in drive 0. Make sure
   it is write-protected for safety.
2. DG DFORMAT
3. Put the new diskette in drive 1.
4. Type: DFORMAT 1
   or DFORMAT 5
   to format the upper or lower surface
respectively.

5. Wait until the LED's on the front panel stop blinking.

6. If there have been no errors, the diskette is formatted and you can now DINIT or DCOPY (remember to DSETUP first if you DINIT)

7. If you want to check that the diskette has in fact been formatted correctly, use the DFETCH command with no arguments after DSETUP'ing the drive with the new diskette. If it says OK, it's good, otherwise try DFORMATing again or try another diskette.

DGET FILENAME
DGET.BAK FILENAME
DGET.OR FILENAME
DGET.XOR FILENAME
DGET.FAST FILENUMBER,FILENAME

Command

gets the FILENAME from the disk. DGET.BAK gets a BAK file. DGET.OR and .XOR do OR's and XOR's respectively when getting screen dumps. DGET.FAST uses the FILENUMBER specified to get the file without searching the diskmap. The FILENUMBER is printed out before the FILENAME in DUSEMAP. It can also be gotten with DLOCK. DGET.FAST will speed up accesses appreciably if you have lots of FILENAMES. Note that once a file has been DPUT, its FILENUMBER remains the same until it is deleted or DPUT again (you can DPUT it twice to preserve the FILENUMBER, by the way). If you specify the wrong FILENUMBER for the FILENAME, it simply doesn't care, so be careful with DGET.FAST!

DINIT MAXNAMECOUNT
DINIT MAXNAMECOUNT,[MESSAGE]

Command

reserves space on a formatted disk, starting at sector 0 (the outermost sector on the disk) for the directory (we call it the DISKMAP) of the contents of the disk. This command initializes the disk, erasing the DISKMAP and making all previous information stored on the disk inaccessible. It works on the currently DSETUP'd drive, and reserves space for the number of entries specified by MAXNAMECOUNT. If specified, the [MESSAGE] is stored so it appears when you look at the diskmap with DUSEMAP; it is a way to state what the diskette is for.
Kinds of entries are: MACROS, ARRAYS, SNAPs, monitor SCREEN dumps, STRINGS, etc.

It is important to plan the initialization of your disk. If you do not plan for enough entries, you may run out of space for names in the diskmap before you run out of actual space on the disk, in which case you will get the "DISKMAP FULL" error message. Likewise, if you allocate too much space for names in the directory, you could be wasting valuable disk space.

To calculate how much directory space should be reserved, use a ratio of 4 entries per sector of diskmap space. Each entry requires 128 bytes to store the entry name, type, size, comments, and pointer to the entry's actual location on the disk. In addition to the 4:1 ratio, allow several sectors for overhead.

For instance, a SCREEN dump (saving on disk all information currently displayed on the monitor SCREEN) uses 16K bytes (or 32 sectors) of a disk. Based on 32 sectors per dump, you can only store 11 screen dumps on one side of a disk. To optimize usable space on the disk, initialize the diskmap for 19 entries, so that 8 sectors are used for the diskmap information and more than 370 sectors remain for storage of screen dumps.

Suppose you will be storing a lot of little strings and macros. In that case, you'd want to have a large diskmap of roughly 300 entries, using almost 78 sectors for the diskmap, leaving about 300 sectors free storage space on the disk.

If you are in doubt about the kind of entries you'll be storing on a disk, a suggested value for MAXNAMECOUNT is 200, which should allow adequate diskmap space and storage space for general purposes.

It is not necessary to initialize a disk (using DINIT) if you use DCOPY, since the initialization information will be copied with the rest of the disk.

Examples:

```
DINIT 300
DUSEMAP ;prints 307 free sectors
DINIT 100
```
DUSEMAP ;prints 357 free sectors
DINIT 20
DUSEMAP ;prints 377 free sectors

See Zgrass Lesson 5 for more information on disk and tape storage.

DISPLAY NAME,XCENTER,YCENTER,DISPLAYMODE,ROTATION
Command takes a SNAPped NAME and writes it at the center indicated using DISPLAYMODE. If not specified, ROTATION is assumed to be 0. Rotation 1 means rotate 90 degrees; rotation 2 means rotate 180 degrees; rotation 3 means rotate 270 degrees. Refer to DISPLAY MODES for the details on the 74 different writing modes. (A SNAPped NAME is actually an ARRAY specially created by the SNAP COMMAND and is essentially an exact copy of an area of screen memory.) You can use DISPLAY for animation. Say there is an apple drawn at the center of the screen which fits inside a rectangle of 48X48 PIXELs. The following code will draw it, SNAP it, and move it on a JOYSTICK.

CLEAR
CIRCLE 0,0,40,1,1
BOX 0,17,4,8,3
SNAP APPLE,0,0,48,48
LEAVE EXTRA WHITE AROUND FOR ERASING
MOVE=[DISPLAY APPLE,X=X+$X1,Y=Y+$Y1,0
SK -1]
MOVE

Note: The largest square area you can SNAP in one piece is 125X125 PIXELs (or about 15625 PIXELs or 1/4 of the screen.)

DISPLAY.SCREEN 0-15,XCENTER,YCENTER,DISPLAYMODE,ROTATION
Command same as DISPLAY but uses contents of the specified SCREEN (0-15) to DISPLAY on the current SCREEN as specified by $MW (see DEVICE VARIABLES) instead of a SNAP.

DISPLAY.PAN NAME,XCENTER,YCENTER,DISPLAYMODE,ROTATION
Command is the same as DISPLAY except that it uses the contents of the specified "super snap" panorama name to display on the current write screen within the current window.
DISPLAY MODES

Idiosyncrasy

the possible values for DISPLAY MODE are between 0 and 159. You may need to study your truth tables for PLOP, XOR, OR, AND, and PRIORITY logical operations to really understand what's going on. There are 10 logic modes, mentioned above, which we combine with 16 filters (0,10,20,...,150) to come up with 160 DISPLAY MODES:

0,1,2,3,4,5,6,7,10,11,12,13,14,15,16,17,...,159

Logic MODES

Meaning:

0 PLOP the SNAPped NAME on the screen
1 XOR the SNAPped NAME with the screen
2 OR the SNAPped NAME with the screen
3 AND the SNAPped NAME with the screen
4 PRIORITY WRITE
  red(01)covers white(00)
  green(10)covers white and red
  blue(11)covers white, red and green
5 PLOP SNAP only on the screen colors mentioned in the filter
6 XOR SNAP only with the screen colors mentioned in the filter
7 OR SNAP only with the screen colors mentioned in the filter
8 AND SNAP only with the screen colors specified by the filter
9 PRIORITY WRITE SNAP only with the screen colors specified by the filter

FILTERS: DISPLAY only this COLOR in SNAPped NAME:

0 everything
10 white (00)
20 red (01)
30 green (10)
40 blue (11)
50 red and blue
60 green and blue
70 red and green
80 white and blue
90 white and red
100 white and green
110 white, red and green
120 white, red and blue
130 white, green and blue
140 red, green and blue
150 display to the nearest 4 pixels
(DISPLAYMODE 150 is special and esoteric. It PLOP's to the screen in groups of four pixels instead of worrying about single pixel boundaries, which makes it about twice as fast as the normal PLOP. Use it if speed is more important than having a snap width not evenly divisible by four.)

The equation for figuring out a specific DISPLAY MODE is:

\[ \text{DISPLAYMODE} = \text{LOGICMODE} + \text{FILTER} \]

---

**DLOAD**
**DLOAD.ZAP**
**DLOAD.CLEAR**
**DLOAD.SET**

**Command**

DLOAD takes the current disk and loads into SCREEN MEMORY, screens 4-15. Then, all references to that DISK will be done from MEMORY.

DLOAD.CLEAR disables the DISK in MEMORY without writing it out. DLOAD.ZAP copies what is loaded into SCREEN MEMORY onto the disk in the currently DSETUP'd drive. DLOAD and DLOAD.ZAP are a good way of making lots of copies of the same floppy disk—just DLOAD the master and then switch disks and do a DLOAD.ZAP for each copy. Of course, if you wish to preserve any changes you have made to the DLOAD'd information, you must use DLOAD.ZAP to write the SCREEN MEMORY back out onto a diskette. Also, you can DINIT screens 4-15 for later DLOAD.ZApping if you DLOAD.SET it first. DLOAD.SET forces screens 4-15 to think they were DLOAD'd from the current drive. It is useful if the system or you somehow cancelled a DLOAD.

---

**DLOOK(FILENAME)**

**Esoteric Function**

returns the FILENUMBER of the FILENAME as indicated by DUSEMAP or -1 if the FILENAME is not in the diskmap. The FILENUMBER is used by DGET.FAST to access a file without searching through the diskmap for the FILENAME.
DOWNER(NUMBER)
Esoteric Function
returns the name of the file associated with the SECTORNUMBER indicated. It is useful for telling which file name is having trouble being read during a DCOPY.

DMATCH(STRING)
DMATCH(STRING,TYPE)
Esoteric Swap Function
uses same syntax as the MATCH function for strings to match names in the disk map (or current submap). DMATCH returns the matched name as a string or a null string if no match is found. Each time you do a DMATCH, it will resume looking in the directory where it left off. DSETUP resets the matching to the first name in the disk map. The optional type allows you to match only certain file types (see WHATSIS for types; screen dumps are type 38).
Examples:

DEEZ=(DS 0
ABC=DMATCH([D*])
IF ABC=(),PR ABC;SKIP -1
the above will print all disk file names on disk 0 starting with D.
PRPIX=(DS 0
ABC=DMATCH([A-Z]*,38)
IF ABC=(),PR ABC;SKIP -1
the above will print all disk files on disk 0 that are screen dumps.

DPUT NAME,[MESSAGE]
DPUT NAME
DPUT.TV NAME,[MESSAGE]
DPUT.TV NAME
Command
puts NAME out on the disk with the message indicated. Messages can be any string and are used for documentation only. DUSEMAP shows them. If there is already a file with the same name and type, the message can be omitted and the old message will be copied over. (If the types don't match, you will get error #81 indicating it.) DPUT automatically creates BAK files (which you can get at with DGET.BAK and delete with DDELETE.BAK or DBAKS). If a BAK file is present already, it is automatically deleted when you DPUT again. DPUT.TV must be used to put out a screen dump.
A common error is to try to DPUT two macros in the same command (as you can with DGET or DDELETE); this causes the first 50 characters of the second macro to be used as the message since Zgrass can't tell the difference between macros and messages (they are both strings). If this happens, DUSEMAP will look strange and you should simply DPUT both macros out again with proper messages.

DRENAME OLDNAME, NEWNAME, [NEW MESSAGE]
Swap Command
renames the oldname to the newname on the disk with the new message.
Example:

DRENAME MANKIND, PERSONKIND, [A NEW MESSAGE]

DSETUP DISKNUMBER
DSETUP DISKNUMBER, SUBMAPNAME
Command
DSETUP does several things. First, it sets the disk to be the "current" one; that is, the one referred to automatically by most disk commands. 0 and 1 are the upper sides of the disks in the drives marked 0 and 1 respectively. 4 and 5 are the lower sides of 0 and 1 respectively. If you are lucky enough to have two double disk drives, the numbers of the second ones are 2 and 3 (upper) and 6 and 7 (lower). If you are even luckier and have a Winchester disk, it is configured as -1, -2, -3, up to -29. DSETUPs of disk numbers between 8 and 127 are for use by special swap modules for as yet unspecified disk drives or pseudo-disk drives. (Esoteric note: DSETUP also causes DMATCH to start looking from the first name in the disk map.)
Second, if the SUBMAPNAME is supplied, the disk commands are all directed to reference only file names within the indicated submap. (DGET will look at the normal disk map after a match failure in the current submap, however). You cannot get a file from another submap nor put a file out into another submap without changing the submapname with DSETUP.

DUSEMAP
DUSEMAP FILENAME
Command
lists all the names on the disk (under the current submap, if any.) If a FILENAME is specified, just that name's map information is printed. The
number printed out at the beginning of each entry is the FILENUMBER which can be used by DGET.FAST to speed up DGETs in time-critical applications.

DZAP SECTORNUMBER,BYTENUMBER
DZAP SECTORNUMBER,BYTENUMBER,VALUE
Esoteric Swap Command/Function
like ZAP but works on disk information. The disk is formatted into 384 sectors of 512 bytes each. Sector 0 holds a byte map indicating used sectors and sectors 1–n have the disk map information. Sectors n+1 through 383 have data. This is a dangerous command since you can permanently confuse a disk if you DZAP it unskillfully. There is more documentation on the disk formats in the Swap Module Documentation, a separate package.
Example:

```
PRBYTEMAP=[A=0;K=0
IF K#255,PR K=DZAP(0,A=A+1);SKIP 0]
```
the above will print out the bytes in sector 0 of the disk until a -1 byte is seen. The zero bytes represent free sectors and the one bytes mark used sectors.
EDIT NAME
Command
edits the MACRO specified.

EDIT CONTROLS:

- **Left Arrow**: Move cursor left
- **Right Arrow**: Move cursor right
- **Up Arrow**: Move CURSOR up
- **Down Arrow**: Move CURSOR down
- **TAB**: Delete line
- **NEXTLINE**: Insert a line
- **ESC**: Delete a character
- **HOME**: Insert a character
- **CTRL+S**: Set copy/move pointers
- **CTRL+T**: Clear copy/move pointers
  
  Also, repaint screen
- **CTRL+D**: Move
- **CTRL+F**: Copy
- **CTRL+E**: Update and exit from editor
- **BREAK**: Exit editor without updating

Note that there are only 80 characters visible on a line. More are permitted if you insist, but you need to split the line by inserting a NEXTLINE character to see or edit those past 80.

ELLIPSE ANGLE, Xcen, Ycen, Xsize, Ysize, TYPE, COLOR MODE
Swap Command
draws an ellipse centered at Xcen, Ycen with Xsize as the width and Ysize as the height in the specified COLOR MODE. Set TYPE to 1 to get a solid ellipse, and 0 to get just the outline. ANGLE is the tilt off the X-axis in DEGREES, unless you tell the system to use RADIANS by setting $RD to 1.

Examples:

- ELLIPSE 0, -50, 50, 80, 40, 1, 1
- ELLIPSE 0, 50, 50, 80, 40, 0, 1

The first draws a solid ellipse, the second just its outline.

- ELLIPSE 45, 50, -50, 80, 40, 1, 2
draws a solid ellipse tilted off the X-axis at 45 degrees.

ERROR NUMBER
Idiosyncrasy
an ERROR NUMBER is printed on the CRT if something has gone wrong in your MACRO, or if you try to do something like dividing by zero which is not
allowed. The ERROR NUMBER is also put into $ER.
Refer to the following list for a clue to what went wrong:

ERROR #: Explanation:

2 ;System error - RESTART system
3 ;System error - RESTART system
4 ;System error - RESTART system
20 ;Operand (VARIABLE, Number, etc.) expected but not seen
21 ;Something other than a legal NAME on the left side of an ASSIGNMENT
22 ;Can't do this conversion, only strings and numbers may be converted to each other
23 ;Arithmetic overflow (number too big to convert to INTEGER or exceeds FLOATING POINT range)
24 ;You tried to divide by zero
27 ;Out of memory space, DELETE something
28 ;More than 128 characters typed before a NEXTLINE
30 ;Too many ARGUMENTs for this COMMAND
31 ;Funny SYNTAX
32 ;Extra stuff on line
33 ;Illegal character after COMMAND name
34 ;This NAME should be a MACRO but it isn't
35 ;Can't find this NAME
36 ;More RETURNS than MACRO calls
37 ;Can't find this LABEL
38 ;This NAME can't be DELETED for system integrity reasons
39 ;Not enough ARGUMENTs for this COMMAND
40 ;No such COLORMODE or DISPLAYMODE
41 ;Illegal character in NAME (must be a followed by letters or digits)
42 ;Unbalanced parentheses
43 ;Number expected but you forgot it!
45 ;This NAME already exists
46 ;Illegal special VARIABLE NAME
47 ;ARRAY reference out of bounds
48 ;More than 4 dimensions specified in ARRAY COMMAND
50 ;No such SWITCH with this COMMAND
51 ;Fraction too small (arithmetic underflow)
52 ;Invalid ARGUMENT value (example: SQRT(-1))
53 ;EDIT only works on MACROS (STRINGs)
54 ;Only A-Z allowed in CONTROL COMMAND
55 ;Too many digits after decimal point
(6 maximum)
56 ;Negative value not allowed here
57 ;Null STRING not allowed here
58 ;Negative ARGUMENT not allowed here
59 ;Can't COMPILE this COMMAND
60 ;Duplicate LABEL
61 ;INTEGERS only for COMPILed SKIPS
62 ;Too many lines for COMPILER
63 ;Illegal LABEL SYNTAX
64 ;ONERROR in LOOP
65 ;LOOPMAX exceeded
66 ;System STRING error
67 ;Too many ARGUMENTS
68 ;Must be in MACRO for this COMMAND
69 ;Can't CMPARA ARRAYS of different sizes
70 ;Transmit error over auxiliary RS232 Port
71 ;Disk Byte map messed up
72 ;No such file
73 ;Feature not implemented
74 ;Disk error
75 ;Too many SKIPS, GOTOs, IFs to
76 ;COMPILE (max is 99)
77 ;Disk full
78 ;Disk track seek error
79 ;Disk read error
80 ;Disk write error
81 ;Can't back up one file type over
82 ;another type which have the same name
83 ;Disk not inserted properly
84 ;Use DDSMAP to delete an entire submap
85 ;A disk in already DLOAD'd. Can't DLOAD
86 ;unless DLOAD.CLEAR is done first
87 ;Must be submap for DDSMAP
88 ;Too many turns in PATTERN or
89 ;PATTERN.FILL
90 ;FIXED POINT stack underflow
91 ;FLOATING POINT stack underflow
92 ;The first argument of the STRIPE
93 ;ranges from 0 to 15
94 ;Can't DLOAD.ZAP unless a disk is DLOAD'd
95 ;Can't DPUT or PUTTAPE compiled macros
96 ;Can't specify a zero dimension in ARRAY
EXCLUSIVE OR
Buzzword

See XOR.

EXECUTE
Buzzword

is computer talk for doing a COMMAND, MACRO, or ASSIGNMENT. It has nothing to do with killing anything. (See CALL)

EXP(N)
Function

returns the value of e (2.71828) raised to the power N.
Examples:

PR EXP(2)
prints 7.38905
PR EXP(1)*EXP(1)
prints 7.3891

EXPRESSION
Idiosyncrasy
is:

1. a CONSTANT (12, 'foo', for example)
2. a NAME (TOM, $X1, POOHBAH, for example)
3. a combination of OPERATORS and CONSTANTS or VARIABLES (+6, -B, -ABC, FF+1, 'tom '& 'sam', Beer*4, for example).
4. a FUNCTION or MACRO call (SIN(a)+COS(b)), MAX(k,F+E,Beer), etc).

Expressions can be simple or complex. Actually, anything syntactically correct in ZGRASS is an EXPRESSION. Arithmetic EXPRESSIONs result in numbers being generated and are a mix of arithmetic OPERATORS (+,-,\,*,?,&,&,|), parentheses, numbers, and VARIABLES. STRING EXPRESSIONs are a mix of STRING OPERATORS ("",',[,],{,},&,@,?) and STRING VARIABLES. FUNCTIONS which return NUMBERS or STRINGS can also be parts of EXPRESSIONs. ZGRASS attempts to convert NUMBERS to STRINGS and STRINGS to NUMBERS when it can, so a STRING like ABC='1234' can legally be used in PRINT ABC+ABC or PRINT ABC&ABC, and so on. COMMANDs are EXPRESSIONs too. Most return the value 1 but some, like ANYARGS, SINE, RETURN, can return other values as well. The basic idea is to combine small EXPRESSIONs to make larger ones. Examples:

A
A+1
A+B*C
(A+B)*c
.F
Esoteric Switch
is a way of telling a MACRO to EXECUTE every 1/60 second. Such MACROs should be short since they take precedence over regular and .B MACROs.
Example:
```
TIMESUP:=[timer=timer+1
   IF timer==180,PRINT '3 SECS ARE UP';timer=0]
TIMESUP.F
```
Unfortunately, unless COMPILED, this takes about 6.2 seconds to do. See TIMEOUT.

FILES
Buzzword
is what things stored on disk or tape are called. FILENAMEs are the NAMEs of FILEs, of course. Never use abbreviations for FILENAMEs!

FLOATING POINT
Buzzword
is computer talk for numbers bigger than 32767 and smaller than -32768 (16 BIT INTEGER range). Numbers outside this range and those with decimal points must be stored and computed specially for esoteric computer reasons. The trade-off is that the range of the numbers available for FLOATING POINT calculation becomes enormous, but the accuracy starts to slip after a while. Fractions are always converted to FLOATING POINT. The name, by the way, comes from the decimal point floating around according to the POWER of ten the number has to be raised to in order to print it out to six digits of accuracy. It is also called 'scientific' notation; and, if you are not a scientist or engineer, you will probably not need to worry about it. You can convert to whole numbers with the INT FUNCTION.

FONT STRING,ARRAYNAME,SNAPNAME,YOFFSET,LEFTX,RIGHTX
Esoteric Swap Command
is used to create and maintain ARRAYS of characters or symbols to be used with the TEXT COMMAND. Each time it's used, the FONT COMMAND adds one character (or symbol) to a FONT ARRAY if it has not been previously defined in that ARRAY or replaces it if it has.
The ARGUMENTs are:

SIN(ABC)+COS(ABC)  
C=A+BOX(10,10,20,30,5)  
etc.
STRING is a single character. This character is used to identify this entry in the ARRAY. When this character is used in a STRING in the TEXT COMMAND, the corresponding character or symbol in the 'SNAPNAME' is displayed on the screen.

ARRAYNAME is the NAME of the FONTARRAY. If this NAME already exists, the character and SNAP are added to it, replacing a previous entry having the same identifying character, if necessary. If the NAME doesn't exist, it's created.

SNAPNAME is the NAME of the SNAP to be copied into the FONTARRAY. This can be a SNAP of any character or symbol, of any size or COLOR. If it is really large, you can't have many in one FONTARRAY before you run out of space.

YOFFSET is a number used along with the Y-COORDINATE in the TEXT COMMAND to determine the Y-COORDINATE used in displaying this character. A negative number drops the character below the line of text, a positive number raises it. This option is used for characters such as lower case g or p, which should drop below the line of text or superscripts which should go up some.

LEFTX and RIGHTX are numbers from 0 to 4. They identify the type of the left or right edge of a character. The type of the right edge of one character, and the left edge of the next, are used in the TEXT COMMAND to look up a horizontal spacing value in a two-dimensional ARRAY.

For example:

LEFT

\ 0 / \ 1
\ 0 / 1 / 2 / 3 / 4 ;

-------------------------------

RIGHT 0 ; * ; * ; * ; * ; * ;

-------------------------------

\ 0 / 1 / 2 / 3 / 4 / 5 \n
-------------------------------

\ 2 / * ; 3 / 4 / 5 / 6 \n
-------------------------------

\ 3 / * ; 4 / 5 / 6 / 7 \n
-------------------------------

\ 4 / * ; 5 / 6 / 7 / 8 \n
The value found represents a number of pixels. This value, along with the horizontal spacing
constant given in the TEXT COMMAND, are used to determine the horizontal spacing between characters. The TEXT COMMAND has a built-in ARRAY with all entries of zero. Users may create their own ARRAYS and use them in the TEXT COMMAND to override the built-in ARRAY. A zero in the row column means use the default spacing. Typically you would define less space between two o's than between two l's or two m's.

Example:

<table>
<thead>
<tr>
<th>Char</th>
<th>LEFTX</th>
<th>RIGHTX</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Y</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Here, the spacing between "L" and "C" would be 3 and the spacing between "C" and "L" is 7.

**FRAME BUFFER**

Buzzword

is used to store the images on the screen. Each PIXEL on the screen is represented by 2 BITS at a location in MEMORY. Changing that MEMORY location will change a specific PIXEL on the screen. There is 16K of screen RAM which means the FRAME BUFFER in ZGRASS has a RESOLUTION of 320 by 201 with 2 BITS per PIXEL.

**FUNCTION**

Buzzword

is a COMMAND or MACRO that returns a value and is used as part of an EXPRESSION. Actually all COMMANDs and MACROs return values of 1 unless something else is specifically returned. Lots of programming languages use the term FUNCTION so we use it here as a gesture towards programmer solidarity.

Examples:

```
GREED=SIN(AVARICE)
FUNNYARRAY(MAX(A,B,C)=-9999
```

MAX is taken as a user defined FUNCTION which returns the largest of three numbers. See the RETURN Command for how MAX is written.
GETERROR(NUMBER)
Esoteric Swap Function
if NUMBER=0, returns the ERROR number that last occurred. Usually used in conjunction with ONERROR to figure out programatically what ERROR condition arose. Cannot be used outside of the MACRO in which the ERROR occurred.
if NUMBER=2, returns the COMMAND line in ERROR as a STRING. It can be used in conjunction with GETERROR(0) to pinpoint the part of the COMMAND in ERROR and point it out friendly-like to the user of your MACRO.
Example:
BAD=[ONERROR 1
  BOX 0,0,"!",1,3
  PRINT "OK"
  RETURN
  1 PR "ERROR #"&GETERR(0),"ON LINE:",GETERR(2)
  RETURN]
will catch the ERROR ("!") is an invalid INTEGER and print out:
ERROR #22 ON LINE: BOX 0,0,"!",1,3

GETTAPE FILENAME
Command
gets the FILENAME from tape. May be a MACRO, ARRAY, or a 16K screen dump (see PUTTAPE). When you
GETTAPE FILENAME
you get a complete directory listing of everything else which is on the tape before FILENAME. You can also see your file being read in by looking at the lights above the switches. If a read error occurs, the next copy will be read (see PUTTAPE). Use the red RST button to prematurely stop GETTAPE.
Example:
GETTAPE FOOD
will search through the tape until it finds FOOD, then print out:
STRING NAME: FOOD
LENGTH: NUMBER (IN BYTES)
A DESCRIPTIVE MESSAGE ABOUT THE FOOD
If it reads a copy of the file which has errors it will print out '***BAD READ***', and look for another copy.
Switches:
.ERR accept the file even if an error is read
.ANY get the next file, whatever its NAME is, on tape and read it in with the NAME
you specify
.OR OR's the screen if doing a screen
dump read
.XOR XOR's the screen if doing a screen
dump read

GOTO LABEL
Command
causes the line which begins with LABEL to be
EXECUTed next. LABELs begin with numbers.
Examples:
These are valid LABELs:
10
1NOW
2small
30000
Example:
SQUARES=[A=80
1AGAIN BOX 0,0,A,A,A/10
IF (A=A-10)>0,GOTO 1AGAIN
PRINT "IS THIS ART?"]

HELP
Command
gives a list of the resident COMMANDs and
FUNCTIONs available along with their ARGUMENTs and
switches.

IF CONDITIONAL,COMMAND
Command
if the CONDITIONAL is satisfied the COMMAND
following is EXECUTed. Otherwise, control is
skipped to the next line. A CONDITIONAL is a
EXPRESSION which evaluates to 0 (false) or 1
(true). Expressions using RELATIONAL OPERATORs
evaluate to true or false, and the rest of the
line (including ';')s is EXECUTed if the condition
is true. Anything that evaluates to 0 or 1 can be
used as part of an IF statement. Note that IF
must always be followed by a space.
For example:
IF A==10,PRINT A; will print the value of A
if it is equal to 10
FIXUP=[PR "I'M YOURS"]
IF 1,FIXUP;.this will always happen
IF FLAG,B=C+D;.this will happen if FLAG==1
IF SIN(BRADIAN)*1.25<=.7,DRAW
The last example shows that complex expressions are allowed in an IF statement. Note: "equals" as a RELATIONAL OPERATOR is ",="", and a single "=" is the ASSIGNMENT OPERATOR, even in IF statements. For example:

```
IF A=B, FOO
EXECUTES FOO only if B#0.
```

INDEX
Buzzword

is the NUMBER indicating which ARRAY element is being picked. The INDEX in ABC(4) is 4. ARRAYS can have multiple indices if they are multidimensional; for example, CHECKERBOARD(8,8), which has indices (0,0), (0,1), ..., (7,7) allowing 64 elements.

INDIRECTION
Buzzword

allows one NAME to hold another NAME as a STRING to be used as a reference. '@' is the indirection OPERATOR. Examples:

```
TOM=12
SAM="TOM"
PRINT @SAM
```

this prints 12

```
MKARRAY=[PR "ARRAY NAME PLEASE"
INPUT.STR ANAME
PR "HOW MANY ARRAY ELEMENTS?"
INPUT n
CLEAR.CRT
ARRAY @ANAME,n
PRINT ANAME, "HAS ELEMENTS 0 TO", n-1
TMP=@ANAME
i=0
PROMPT ANAME&"("&i&")=?"
INPUT q
TMP(i)=q
IF (i=i+1)<n, SK -3
PRINT "ARRAY", ANAME, "HAS THE VALUES:"
```

```
i=0
PRINT ANAME&"("&i&")="&TMP(i)
```

When you EXECUTE MKARRAY, first it makes an ARRAY with ANAME of size n, then the user inputs values for each ARRAY element, and finally the contents of the ARRAY is printed out. The detail to notice is:

```
TMP=@ANAME
```

This is a shortcut for dealing with ARRAY elements.
in a general program, so that each element can be accessed as TMP(0), TMP(1),...,TMP(n). We could skip the assignment of @ANAME to TMP and instead build a string:

\[ \text{@ANAME}&("&1&")\]

which is the same as

\[ \text{TMP}(1) \]

Unfortunately, the building of strings through CONCATENATION is time-consuming.

**INFINITE LOOP**

Buzzword

is a LOOP which has no intention of ever stopping. Such a LOOP is an error if you want the MACRO it's in to stop or are using it as a FUNCTION which is supposed to return a value. It can be useful, though, as a MACRO run under .B or .F mode or something you want to get out of by using CTRL+C. The LOOPMAX COMMAND can be used to catch infinite loops.

**INPUT NAME1,NAME2,...,NAMEN**

Command

gets the VALUE from the user or the ARGUMENT list passed to the MACRO and stores the VALUE as a number in NAME.

Examples:

ABS=[INPUT a
    IF a<0,RETURN -a
    RETURN a]

PRINT ABS(-10)

prints out 10

PRINT ABS(10)

prints out 10

ASK=[PROMPT "WHAT'S YOUR AGE?"
    INPUT AGE
    PRINT "YOU ARE",AGE*12,"MONTHS OLD AT LEAST"]

if EXECUTED by typing:

ASK 33

the PROMPT is suppressed.

if EXECUTED by typing:

ASK

the PROMPT is printed, and you have to supply the ARGUMENT by typing it in.

Note: if you are passing a VARIABLE (rather than a number, as above), make an EXPRESSION of it by adding 0 or using the "?" OPERATOR so its VALUE is passed rather than its NAME. This is particularly important when passing LOCAL VARIABLES and ARRAY references.
INPUT.NAME NAME

Command

gets a STRING of characters from the user or the ARGUMENT list passed to the MACRO and checks it for valid SYNTAX, and then puts it into NAME as a STRING.

Example:

WHO= [PROMPT "TYPE YOUR FIRST NAME:" 
      INPUT.NAME NAME1 
      PRINT NAME1,"IS A FUNNY NAME!"]

Note: Do not use INPUT.NAME to pass VARIABLES to called MACROS if it is the value of the VARIABLE you want to pass. Use INPUT.STR to pass a STRING in a VARIABLE to a called MACRO.

INPUT.STR NAME1,NAME2,...,NAME N

Command

gets a STRING of characters and then puts it into NAME. This option is good for reading an entire line from the terminal, including commas. It must also be used to pass a STRING with commas or spaces as an ARGUMENT, in which case it should be enclosed in quotes or other STRING delimiters.

Examples:

MAILINGLIST=[PROMPT "TYPE IN A NAME, ADDRESS, AND PHONE # FOLLOWED BY A BLANK LINE" 
             CR={ } 
             PROMPT "MORE:" 
             INPUT.STR INFO 
             IF INFO{},LIST=LIST&INFO&CR;SK -2]

Note: when passing LOCAL STRING VARIABLES to MACROS, make EXPRESSIONs out of them by CONCATENATING them with a null string or by using the "?" OPERATOR in front of the NAME so that the VALUE of the STRING is passed rather than the NAME of the STRING.

If you should want to pass both numbers and strings together to a macro, use INPUT.STR for the strings and regular INPUT for the numbers; do not mix them.

Example:

DOIT GEORGE,14,21,3.3 
GEORGE=[INPUT.STR WHO; .GET THE STRING 
       INPUT X,Y,Z; .GET THE NUMBERS 
       ;.and so on]

(Esoteric note: if you accept everything as
strings and then use some of them later as numbers it will work but the system overhead is greater and you may perceive that it is slower.)

**INT(NUMBER)**

*Function*

FUNCTION which returns the INTEGER part of a number. INT(5.8) will give 5, 6, or 7 without the fractional part, for example. INT truncates the number. If you want roundoff, add .5 first as in INT(X+.5).

**INTEGER**

*Buzzword*

An integer in ZGRASS is a number between 32767 and -32768. It is very easy for the computer to store and deal with numbers in this range so they are used often. Fractions and decimal points are not allowed in INTEGER arithmetic.

**INTERRUPT**

*Esoteric Buzzword*

The ZGRASS System is programmed to EXECUTE a chunk of special code every 1/60 of a second, when the code is "interrupted" by the vertical sync of the TV scan. .F causes a macro to run every 1/60 second.

**ITERATION**

*Buzzword*

is the process of solving things by doing LOOPS. Typically, in computing, ITERATION means doing things incrementally. For instance, a computer would probably walk over to the wall by accurately measuring the distance between it and the wall, computing the exact number of steps needed, and then it would take a step, see if all the steps it had to take were taken yet, and take another if not. If it made a mistake, it might crash into the wall. People, of course, do things through feedback, and often you can program that way with computer systems that are significantly better connected to you than the average payroll-check Stamper (like ZGRASS is, of course). To draw 100 RANDOM sized BOXes on the screen, you could type in 100 different BOX COMMANDs, or write a MACRO which would do it. For example:

```plaintext
SQUARES=[B=0
  BOX -150%150,-90%90,1%50,1%30,1%8
  IF (B=B+1)<100,SK -1]
```
JOYSTICK
Idiosyncrasy
is the gadget with the knob and the trigger that is connected to the ZGRASS machine. You can have up to four joysticks. The first one's knob is known as $K1, its X value as $X1, its Y value as $Y1, and its trigger value as $T1 (see DEVICE VARIABLES).

JUMP ADDRESS
Esoteric Command
Refer to the Swap Module creation documentation, a separate package.

JUMP.ERR STRING
Esoteric Command
causes the STRING to be executed if an error is detected. The error number is stuffed into $ER and all the actions of a CTRL+C are taken before the STRING is executed. JUMP.ERR is good for catastrophic or unexpected error recovery in user-oriented programs.

LABEL
Idiosyncrasy
GOTO 1THIS causes ZGRASS to move to whatever line begins with the LABEL 1THIS. LABELs in ZGRASS start with numbers to differentiate them from NAMEs which cannot start with numbers. LABELs also cannot contain punctuation. You can't have one GOTO in a MACRO go to a LABEL in another MACRO.

Note: lines beginning with a LABEL cannot be indented. Often people leave the lines beginning with LABELs over at the left margin and indent the rest of the lines to make the macro clearer to follow. This takes a small amount of extra space, but is highly recommended.

LEN(STRING)
Esoteric Function
returns the length of a character STRING. If the ARGUMENT is a null STRING, 0 is returned.

Example:
PRINT LEN("abcdef")
prints the VALUE 6
LEN, NUM NUMBER
Esoteric Function
makes the system print out numbers to NUMBER
(range 0-6) decimal places. The default is 6.
Example:
LEN, NUM 2
PRINT 1.41/3
prints 1.33

LINE XCOORDINATE, YCOORDINATE, COLORMODE
Command
draws a line from the previous line endpoint used
in the current MACRO to the endpoint specified by
the XCOORDINATE and YCOORDINATE in the COLORMODE
indicated. LINE X,Y,4 will move the endpoint
without drawing anything and can be used to set
the first endpoint if you do not want the first
LINE to start at (0,0). See COLOR MODES. Each
MACRO has its own place to store the last endpoint
used and it is set to zero when the MACRO is
called.

Note: if you look carefully, you can see that
lines always draw in one general direction, not
really from the last endpoint to the new one. If
this were not done, the line drawn from (-50,20)
to (20,40) for instance, would look different
depending on which endpoint were given first.
This is just another ugly artifact of trying to
draw a nice diagonal on the coarse grid that is
your Zgrass screen.
Example:
LINE 50,-30,1
draws a line from 0,0 to 50,-30.
LINE -50,20,2
draws a line from 50,30 to -80,20.
LINE 50,50,4
LINE 50,-50,3
LINE -50,-50,3
LINE -50,50,3
LINE 50,50,3
draws a rectangle outline.
ZIGZAG=[LINE -160%159,-100%100,0%15;SK 0]
ZIGZA: will draw RANDOM lines of different COLORS
all over the screen.
LOCAL VARIABLE
Esoteric Idiosyncrasy

a VARIABLE which starts with a lowercase (a-z) letter. LOCAL VARIABLEs are known only to the MACRO they are in and are deleted automatically when the MACRO returns. They help save memory and are really useful in .B, .F, and RECURSIVE MACROs.

Note: to pass the value of a local variable to another macro, use the question mark operator to cause it to evaluate in the proper context: PASSIT(?x,?y), for example.

LOGICAL OPERATOR
Buzzword

returns a truth value (0 or 1). ZGRASS has logical "AND" and "OR". The "AND" OPERATOR is '&&'. The logical "OR" OPERATOR is '||'. They are useful in many situations, one of which is combining conditionals in IF statements. Examples:

BEEPTHEJEEP=[CONTROL 14,1]
IF A==10&&B==20,BEEPTHEJEEP;.done if A is 10 and B is 20
IF A==10||B==20,BEEPTHEJEEP;.done if either is true

LOOP
Buzzword

is a series of COMMANDs done over and over. If the loop never stops, it is called an INFINITE LOOP. LOOPS in ZGRASS are constructed with IF's, GOTOs and SKIPS or with .B and .F. You can always get out of a LOOP with CTRL+C. CTRL+Z allows you to get out of a LOOP to do something and then get back in by pressing the RETURN key. A loop is an example of ITERATION.

Examples:

INFINITELoop=[PRINT A=A+1;SK 0]
is a loop which will not stop because it doesn't have an end condition. CTRL+C will stop it.

LOOPWHICHSTOPS=[A=0
PRINT A
A=A+1
IF A<10,SKIP -2]

LOOPWHICHSTOPS prints 0 through 9 and stops.

LOOPMAX NUMBER
Esoteric Command

allows you to catch INFINITE LOOPS by setting a maximum for the NUMBER of SKIPS and GOTOs that can occur before ERROR #65 is caused. Macros which contain a LOOPMAX command cannot be COMPILED.
Make sure when you use LOOPMAX that you set it up outside the loop or it won't work correctly.

Example:

```
TEST: [CONTROL 1, 0; SET CTRL+T TO ZERO
PRINT "HIT CTRL+T"
ONERROR 1 SLOW
LOOPMAX 100
IF CONTROL(1)#1, SK 0
RETURN
1 SLOW PRINT "YOU DIDN'T HIT CTRL+T FAST ENOUGH!"
```

LN(NUMBER)

Function

returns the natural log of NUMBER.

LPAD(STRING, CHARACTER, FIELDWIDTH)

Esoteric Swap Function

returns a pointer to the STRING, padded on the left with a specified CHARACTER so that it fits within a given FIELDWIDTH.

Examples:

```
PR LPAD("ABC", "\*", 6)
prints out the STRING "\\*\\*ABC"
PR LPAD("EXAMPLE", "\*", 5)
prints out the STRING "AMPLE"
LEFTX=[A=2
PR LPAD(A, 'X', 5)
A=A*10; IF A<=20000, SK -1]
```

takes each VALUE of A and pads it on the left with X's until each number is printed in a field of 8 characters. Usually used with blanks, not X's.

```
XXX2
XXX20
XX200
X2000
20000
```
MACRO

Idiosyncrasy

is a STRING that contains legal ZGRASS COMMANDs. Most programming languages call such things 'programs' or 'subroutines'. MACROS are user-defined COMMANDs. You can pass ARGUMENTs to MACROS with the INPUT COMMAND and return values with the RETURN COMMAND. You define a MACRO just like you define a STRING (with an ASSIGNMENT to a NAME or by using EDIT).

MATCH(OTEXT,MTEXT,LOWER,UPPER)

Esoteric Swap Function

Search for the occurrence of MTEXT, a STRING, within a specified range of OTEXT, another STRING. If a MATCH is found, the returned displacement value is relative to the beginning OTEXT, the first character being the 0th one. -1 is returned if a MATCH was not found within the specified limits. The search for a MATCH may proceed from either direction. If UPPER is greater than or equal to LOWER a forward search is made. If UPPER is less than LOWER a backward search is made. (That is, the characters are still matched left to right bu the pointer backs up on failure to match instead of advancing.) MTEXT does not necessarily have to contain all the characters of the desired MATCH but rather, may use the following expression symbols:

? (wild card) MATCH any one character

* MATCH all characters

*text MATCH all characters preceding actual text

text* MATCH text and all remaining characters following text

text1*text2 MATCH all characters between text1 and text2

[chars] MATCH first occurrence of any one of the characters with the '[',']'s. All the expression symbols lose their special meaning when appearing within square brackets.

[char-char] MATCH any character within the range specified. [0-9] is the same as specifying [0123456789]. The minus sign
loses its special meaning when specified as first or last character within the square brackets.

ignore the following character's special meaning

anchor MATCH to beginning or end of OTEXT depending on whether the anchor symbol occurs first or last within MTEXT

Examples:

<table>
<thead>
<tr>
<th>Example</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR MATCH(&quot;VACATION&quot;,&quot;CAT&quot;,0,7)</td>
<td>2</td>
</tr>
<tr>
<td>PR MATCH(&quot;VACATION&quot;,&quot;CAT&quot;,0,3)</td>
<td>-1</td>
</tr>
<tr>
<td>PR MATCH(&quot;ABABCDAB&quot;,&quot;?A&quot;,0,10)</td>
<td>1</td>
</tr>
<tr>
<td>PR MATCH(&quot;ABABCDAB&quot;,&quot;?A&quot;,4,10)</td>
<td>5</td>
</tr>
<tr>
<td>PR MATCH(&quot;ABABCDAB&quot;,&quot;?A&quot;,10,0)</td>
<td>5</td>
</tr>
<tr>
<td>PR MATCH(&quot;ABABCDAB&quot;,&quot;?A&quot;,4,0)</td>
<td>1</td>
</tr>
<tr>
<td>PR MATCH(&quot;SIGNAL&quot;,&quot;*&quot;,0,20)</td>
<td>0</td>
</tr>
<tr>
<td>PR MATCH(&quot;WHAT TIME?&quot;,&quot;MEV?&quot;,0,10)</td>
<td>7</td>
</tr>
<tr>
<td>PR MATCH(&quot;SIGNAL&quot;,&quot;*&quot;,20,0)</td>
<td>0</td>
</tr>
<tr>
<td>PR MATCH(&quot;SIGNAL&quot;,&quot;*&quot;,3,20)</td>
<td>3</td>
</tr>
<tr>
<td>PR MATCH(&quot;THIS IS A TEST&quot;,&quot;[AEIOU]&quot;,3,7)</td>
<td>5</td>
</tr>
<tr>
<td>PR MATCH(&quot;THIS IS A TEST&quot;,&quot;[A-H]&quot;,15,0)</td>
<td>11</td>
</tr>
<tr>
<td>PR MATCH(&quot;GRAPHICS&quot;,&quot;;&quot;,0,10)</td>
<td>7</td>
</tr>
<tr>
<td>PR MATCH(&quot;COMPUTER GRAPHIX&quot;,&quot;G*X&quot;,5,20)</td>
<td>9</td>
</tr>
</tbody>
</table>

MEMORY

Buzzword

is computer storage which is divided into BYTES. ZGRASS has 320K BYTES of MEMORY. 32K is ROM (Read Only Memory) where the resident code for ZGRASS is stored. 256K is Screen RAM (Random Access Memory that feeds the TV screen). 32K is RAM used to store MACROS, ARRAYs, SWAP MODULEs, SNAPS, and VARIABLES. USEMAP shows usage of the 32K RAM. CORE tells you how much of the 32K RAM you have free.

MMOVE SOURCE,DESTINATION,LENGTH

MMOVE.UP SOURCE,DESTINATION,LENGTH

Esoteric Command

Uses the Z-80 LDIR (block memory move) instruction to move the number of bytes given by LENGTH from the SOURCE to the DESTINATION. It's good for esoteric manipulations of screen memory. Beware, you can also scramble user memory easily. Very Esoteric Note: MMOVE does a LDDR Z-80 instruction. MMOVE.UP does a LDIR Z-80 instruction. The first argument is HL, second DE, and the third is BC.
Examples:

\texttt{NB}
\begin{verbatim}
MOVE 31600,32000,15200
\end{verbatim}

moves image down

\begin{verbatim}
MOVE UP 16384+80,16384,16000
\end{verbatim}

moves image up

\begin{verbatim}
MOVE 31998,32000,15200
MOVE 31919,32000,15000
MOVE 21919,22000,5000
MOVE 31919,32000-16384,15000
\end{verbatim}

The last one shows the use of XOR (works only if XOR set by drawing a box with XOR -- as NB does.) Also works with OR if last box was OR'd. Note that screen source is addressed at 0 by subtracting 16384, which kicks in the special XOR and OR hardware. Note that if $ML$ is 1, you can use $MR$ and $MW$ to do clever copying between screen pages. See SCREEN.

\texttt{NAME}

Idiosyncrasy

\texttt{NB}

is any set of symbols starting with a letter that has a VALUE (TOM=5, SAM="HOWDY", for example) or an ARRAY of VALUES (ARRAY WOMEN,13, for example). A NAME must start with a letter (or '$') and has only letters and numbers (0-9) and '$'s in it. The rule is that a STRING is not a name if it starts with a number. In this case, it is either a NUMBER or a LABEL (LABELs must be the first thing on a line, of course). If it starts with a letter, it is a NAME. Any kind of punctuation ends the NAME. A NAME is also an EXPRESSION, although a very simple one. NAMES joined together with numbers and other NAMES using punctuation (+,-,/,*,(,),etc.) are EXPRESSIONs. If a NAME begins with a lowercase letter, it is LOCAL and is known only to the MACRO in which it occurs. For example, sam=5.

\texttt{NB}

System Macro

draws a bunch of boxes. NB.B will continue forever drawing boxes.
NC
System Macro
draws circles and boxes on all 16 screens and then
does an ND.F

ND
System Macro
contains the statement $TV=$IV+1. ND.F is a good
way to flip through all the screens.

NEXTLINE
Idiosyncrasy
is the code ZGRASS uses to represent the end of a
line. It is generated by the RETURN key. Sometimes it is known as the 'carriage return' or
'CR' from the old days or 'RETURN' on most
keyboards (not to be confused with the RETURN
COMMAND, of course). This character is at the end
of every line in a MACRO except possibly the last.
It is also the key which tells ZGRASS you are
finished typing in the line you have been typing. If
you hit CTRL+Y and then list out a MACRO, you
will see a '!' marking the position of each
NEXTLINE. NEXTLINE also advances the 20-line
printout mode started by CTRL+W.
Note: you cannot have any spaces before the
NEXTLINE. CTRL+Y is good for verifying that no
spaces exist between the last character on the
line you've typed and the NEXTLINE. In edit, also
use CTRL+T.

NUMBER
Buzzword
Examples:
  1778
  1.5
  -44.3
  3.5E6 (3.5 million)
  -2E-9 (-2 trillionths)

NUMERIC VARIABLE
Buzzword
is a VARIABLE which has a NUMBER as its value.
USEMAP will tell you it is a NUMNAME.

ONERROR LABEL
Esoteric Command
sets up a transfer to LABEL when an ERROR occurs.
You can turn off ONERROR by specifying no LABEL
(ONERROR by itself turns the normal ERROR CODES
back on). You normally put a ONERROR LABEL before
a statement that is likely to cause an ERROR. You can only have one ONERROR setup per MACRO at a time, but you can change it in the MACRO anytime. MACROS which have ONERROR commands cannot be COMPILED. See LOOPMAX and GETERROR for examples of ONERROR. Note: this COMMAND precludes you from EXECUTING a MACRO NAMEd "ONE" due to the ABBREVIATION POLICY. This is a common mistake.

OPERATOR

Idiosyncrasy is what glues NUMBERS and NAMEs into EXPRESSIONs. OPERATORs take the values they operate on and return a single value. Each OPERATOR has a precedence, that is, a pecking order for evaluation.

<table>
<thead>
<tr>
<th>OPERATOR</th>
<th>MEANING</th>
<th>PRECEDENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>@</td>
<td>indirect</td>
<td>9</td>
</tr>
<tr>
<td>?</td>
<td>value</td>
<td>9</td>
</tr>
<tr>
<td>-</td>
<td>unary minus</td>
<td>8</td>
</tr>
<tr>
<td>!</td>
<td>absolute value</td>
<td>8</td>
</tr>
<tr>
<td>+</td>
<td>unary plus</td>
<td>8</td>
</tr>
<tr>
<td>%</td>
<td>random</td>
<td>7</td>
</tr>
<tr>
<td>/</td>
<td>division</td>
<td>6</td>
</tr>
<tr>
<td>\</td>
<td>modulus</td>
<td>6</td>
</tr>
<tr>
<td>*</td>
<td>multiply</td>
<td>6</td>
</tr>
<tr>
<td>+</td>
<td>add</td>
<td>5</td>
</tr>
<tr>
<td>-</td>
<td>subtract</td>
<td>5</td>
</tr>
<tr>
<td>&amp;</td>
<td>concatenate</td>
<td>5</td>
</tr>
<tr>
<td>==</td>
<td>equals</td>
<td>4</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
<td>4</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
<td>4</td>
</tr>
<tr>
<td>&lt;= or =&lt;</td>
<td>less than or equals</td>
<td>4</td>
</tr>
<tr>
<td>&gt;= or =&gt;</td>
<td>gr. than or equals</td>
<td>4</td>
</tr>
<tr>
<td># or &lt;&gt;</td>
<td>not equals</td>
<td>4</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>logical AND</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>assign</td>
<td>1</td>
</tr>
<tr>
<td>()</td>
<td>parentheses</td>
<td>0</td>
</tr>
</tbody>
</table>

OPERATORs with higher PRECEDENCE are done before ones with lower PRECEDENCE and ones with equal PRECEDENCE are done from left to right. Examples:

- $2+3*4$ equals $14$
- $(2+3)*4$ equals $20$
- $-7*3+2$ equals $-19$
Ok

Buzzword

works on BITs. It makes BITs OR'ed with 1's equal to 1, and leaves BITs OR'ed with 0's the same as they were.

OR table using 2 BITS:

<table>
<thead>
<tr>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>OR</td>
<td>00</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>===</td>
<td>===</td>
<td>===</td>
<td>===</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>===</td>
<td>===</td>
<td>===</td>
<td>===</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>===</td>
<td>===</td>
<td>===</td>
<td>===</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>===</td>
<td>===</td>
<td>===</td>
<td>===</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>===</td>
<td>===</td>
<td>===</td>
<td>===</td>
</tr>
</tbody>
</table>

The OR COLOR MODEs are 8-11. The OR DISPLAY MODEs are 2,12,22,...,132,134.

OVERFLOW

Buzzword

is what happens when the range of a CONSTANT, VARIABLE, or EXPRESSION is too large or too small. For instance, many DEVICE VARIABLES represent a single BYTE of information which gives a range of 0-255 or -128 to 127. Exceeding this range causes WRAP-AROUND so 256 is actually 0, 257 is 1, 258 is 2. INTEGERS overflow after 32767 and under -32768, which causes ERROR # 23.

PANORAMA

Idiosyncracy

refers to a "super snap" stored in some or all of SCREENs 4-15. See DISPLAY.PAN, POINT.PAN, SCALE.PAN, BUILD AND PLACE.

PATTERN X,Y,XOFFSET,YOFFSET,SNAPNAME

Command

like PATTERN.FILL but uses PIXELs out of a SNAPNAME to fill within a bounded area. X and Y indicate the starting point for the pattern fill. The area is filled with SNAPNAME as if its lower left corner were positioned at 0,0. XOFFSET and YOFFSET are used to change this orientation. The following example illustrates the use of the pattern fill with and without offsets.

Example:

OPART=[CLEAR
CIRCLE -7,0,20,1,1
CIRCLE 7,0,20,1,7
SNAP SQR,0,0,30,20
CIR 0,0,100,1,2
CIR 0,50,40,1,0]
OPART
PATTERN 40,0,0,0,SQR;.NO OFFSET
PATTERN -40,0,14,-15,SQR;.OFFSET ON X AND Y
PATTERN 0,0,0,0,SQR

If there is a hole in the boundary of the area being filled, you will get a flood of color spilling out. In this, or any other case, PATTERN can be stopped with CTRL+E.

PATTERN.FILL X,Y,FILLCOLOR

Command
is used to fill a bounded area with a solid color. X and Y are the coordinates of a point interior to the boundary. FILLCOLOR can be ANY COLORMODE 0-21.

Refer to PATTERN for filling areas with a pattern. Example:
BOX 0,0,80,80,1
BOX 20,20,40,40,0
BOX -20,0,36,76,0
BOX 18,-20,40,36,0

Creates an L shaped area in red ($L1).

PATTERN.FILL 10,-20,3
will fill the area bounded by the red outline with blue starting at the point 10,-20.

PIXEL
Buzzword

is the smallest thing you can change on the screen. The POINT COMMAND will fill one pixel with a COLOR. The screen is divided into 64640 pixels (320*201). There are 320 PIXELs horizontally and 201 vertically. The center of the screen is (0,0) and the PIXELs are numbered -160 to 159 horizontally (X-axis) and -100 to 100 vertically (Y-axis). The POINT FUNCTION will give you the COLOR VALUE of a PIXEL.

Due to the fact that the PIXELs are not quite square, circles are somewhat elliptical and squares are somewhat rectangular on most TV sets; this is a non-adjustable hardware constraint.
PLACE NAME, XVAL, YVAL, XCEN, YCEN, XSIZE, YSIZE, DISPLAYMODE

Esoteric Swap Command

puts the area on the current screen defined by XCEN, YCEN, XSIZ, YSIZ into the "super snap" named at the location XVAL, YVAL. The DISPLAYMODE, if unspecified, is taken to be 0. The expected clipping will take place, of course. Note: PLACE works with DISPLAYMODEs 0,1,2 only.

PLOP

Buzzword

means that whatever COLOR you write with (00-11) will cover whatever is on the screen. So:

Y PLOP X equals Y

For Example:

00 PLOP 10 equals 00
10 PLOP 11 equals 10

PLOP table using 2 BITS.

PLOP with:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>===</td>
<td>===</td>
<td>===</td>
<td>===</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>===</td>
<td>===</td>
<td>===</td>
<td>===</td>
</tr>
<tr>
<td>01</td>
<td>00</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>===</td>
<td>===</td>
<td>===</td>
<td>===</td>
</tr>
<tr>
<td>10</td>
<td>00</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>===</td>
<td>===</td>
<td>===</td>
<td>===</td>
</tr>
<tr>
<td>11</td>
<td>00</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>===</td>
<td>===</td>
<td>===</td>
<td>===</td>
</tr>
</tbody>
</table>

The PLOP COLOR MODEs are 0-3. The PLOP DISPLAY MODEs are 0,10,20,30,...,130,140.

POINT(XCOORD, YCOORD)

Function

returns the value (0-3) of the PIXEL ADDRESSED by the two COORDINATES given. 0 means that COLOR (00) is at that ADDRESS on the screen, 1 means that COLOR (01) is there, etc. If the ADDRESS is outside the current WINDOW area, a -1 is returned.

Example:

BOX 0,0,40,40,1
PRINT POINT(18,15)

prints 1

PRINT POINT(50,70)

prints 0
POINT XCOORDINATE,YCOORDINATE,COLOORMODE

Command
draws a point at XCOORDINATE,YCOORDINATE in the
COLOR MODE specified. A POINT is one PIXEL in
size. See COLOR MODE.
Examples:
POINT 80,30,1
draws a red point at 80,30
POINT 40,20,2
will draw a green point at 40,20.
SPIRAL=[angle=0;radius=0
x=radius*SIN(angle)
y=radius*COS(angle)
POINT x,y,1
angle=angle+18
radius=radius+.5
SKIP -5]
SPIRAL draws a spiral starting at 0,0. Press
CTRL+C to stop it.

POINT.SNAP(SNAPNAME,XCOOR,YCOOR)

Esoteric Function
returns the value (0-3) of a PIXEL in SNAPNAME
ADDRESSed by XCOOR and YCOOR. XCOOR and YCOOR are
relative to the center (0,0) of the SNAP. -1 will
be returned if the PIXEL is outside the SNAP.
Example:
Create a three-color SNAP using the macro PSNAP
PSNAP=[CLEAR
BOX -10,0,10,20,3
BOX 0,0,10,20,2
BOX 10,0,10,20,1
SNAP FLAG,0,0,30,20]
use TEST to find the color value of the PIXELs in
the SNAP FLAG
TEST=[PROMPT"INPUT X,Y POSITIONS IN SNAP"
INPUT x,y
PRINT POINT.SNAP(FLAG,x,y)
SKIP -3]

POINT.SNAP SNAPNAME,XCOOR,YCOOR,COLOORMODE

Esoteric Command
charges the PIXEL at XCOOR,YCOOR in SNAPNAME in
the COLOORMODE specified. XCOOR and YCOOR are
relative to the center of the SNAP which is 0,0.
You have to DISPLAY the SNAP to see the changes,
of course.
Example:
The following MACRO will change any of the PIXELs
in FLAG which are red (COLOR 01) to blue (COLOR
11).
COLORCHANGE=[CLEAR
BOX -10,0,10,20,3
BOX 0,0,10,20,2
BOX 10,0,10,20,1
SNAP FLAG,0,0,30,20
PR"WATCH THE FLAG CHANGE COLORS"
y=(FLAG(1)/2);sy=-y
PR sy,y
x=FLAG(0)/2
sx=-x
c=POINT.SNAP(FLAG,sx,sy)
IF c=1,POINT.SNAP FLAG,sx,sy,3
IF c=2,POINT.SNAP FLAG,sx,sy,1
IF c=3,POINT.SNAP FLAG,sx,sy,2
IF (sx:sx+1)<:x,SK -4
DISPLAY FLAG 0,0,0
IF (sy:sy+1)<y,SK -7]

The X size of FLAG is stored in FLAG(0). The Y size is stored in FLAG(1). This information is used to determine the setup for two nested loops which will go PIXEL by PIXEL through the SNAP FLAG looking for 01 PIXELs and changing them to 11, 10 PIXELs to 01, and 11 PIXELs to 10 PIXELs. After an entire horizontal line of the SNAP has been evaluated, FLAG will be DISPLAYed.

POINT.PAN NAME,X,Y,COLORMODE
POINT.PAN(NAME,X,Y)
Esoteric Command/Function
functions just like POINT but uses the "super snap" named.

PORT(NUMBER)
Esoteric Function
returns the VALUE read at the PORT NUMBER identified.
Example:
PR PORT(20)
will print the value of the switches 0-7. If switches 0,1,2,3 are down, 15 will be printed.

PORT NUMBER1,NUMBER2
Esoteric Command
writes NUMBER2 to the PORT identified by NUMBER1.
Examples:
A=0
COUNT=[PORT 38,A
A=A+1
WAIT 1
SKIP -3]
will cause the lights to count in binary, one per
second until WRAPAROUND occurs after 255.

\[
\text{FASTERCOUNTDOWN} = [\text{PORT 38}, \text{A} \\\text{ZO} = 32767 \\text{PORT 38, ZO} \\text{IF } ZO \neq 0, \text{SK} -1]
\]

ZO is a system timer decremented by 1 every 1/60th second. When ZO hits 0, it is no longer decremented.

PORTS

Idiosyncrasies

are hardware ADDRESSes for DEVICEs and various input and output gadgets. Some are massaged and put into DEVICE VARIABLES (like the JOYSTICKS & COLOR VARIABLES). Some are accessed with COMMANDs (like RS232).

Example:

\[
\text{PRINT PORT(N)}
\]

will print what the value is at PORT N is.

\[
\text{PORT N,K}
\]

will set PORT N to the VALUE in K

OUTPUT PORTS: (write only)

PORT #: FUNCTION:

10 Vertical Blanking Line
12 Magic Register
16 Master Oscillator
17 Tone A Frequency
18 Tone B Frequency
19 Tone C Frequency
20 Vibrato
21 Tone C VOLUME and Noise Modulation Control
22 Tone B Volume and Tone A Volume
23 Noise Volume
25 Expand Register
38 Lights 0-7 (Bit 0=Light 0)
39 Lights 8-15 (Bit 0=Light 8)
40 Controls Tape Motor Switch 1=on 0=off
\[
\text{PORT 40,1 turns on Motor1} \\\text{PORT 40,2 turns on Motor2} \\\text{PORT 40,3 turns both on} \\\text{PORT 40,0 turns both off}
\]
INPUT PORTS: (read only)

- Bit values for each Joystick
  - Bit 0: UP on (Y)
  - Bit 1: Down on (Y)
  - Bit 2: Left on (-X)
  - Bit 3: Right on (X)
  - Bit 4: Trigger
  - Bits 5-7 not used

16. Joystick 1 ($X1,$Y1,$T1)
17. Joystick 2 ($X2,$Y2,$T2)
18. Joystick 3 ($X3,$Y3,$T3)
19. Joystick 4 ($X4,$Y4,$T4)
20. Switches 0-7 (bit 0=switch 0)
21. Switches 8-15 (bit 0=switch 8)
28. Knob Joystick 1 ($K1)
29. Knob Joystick 2 ($K2)
30. Knob Joystick 3 ($K3)
31. Knob Joystick 4 ($K4)
46. Tablet data

INPUT/OUTPUT PORTS: (read and write)

32. Terminal RS232 data
33. Accessory RS232 data
34. Terminal RS232 control
35. Accessory RS232 control
36. NCUDAT (9511 chip) data port
37. NCUCOM (9511 chip) status port
41. Tape Data Bit 0=Data

POWER(NUMBER1,NUMBER2)
Function

- Returns NUMBER1 raised to the POWER of NUMBER2.
Example:
  PRINT POWER(5,3)
  prints 125

PRECEDENCE
Buzzword

- Is the pecking order for the evaluation of OPERATORS in EXPRESSIONs. See OPERATOR for the PRECEDENCE order in ZGRASS.

PRINT THING
Command

THING (a NUMBER, ARRAY VALUE, EXPRESSION etc.) is converted to a STRING, if possible, and printed followed by a NEXTLINE. Several STRINGS can be used. If you separate them by commas, a space is printed between them. If you do not want the space, separate them with &'s. Stuff in quotes
can also be used (like PRINT "THE ANSWER IS:",A).
PRINTS (and PROMPTS) are suppressed if there are
ARGUMENTs passed to the MACRO.
Examples:
  PRINT 5
  will print 5
  A=1;B=333
  PRINT A&B
  will print 1333
  ME=7
  PRINT 5,ME
  will print 5 7
  PRINT "A"&"B"&"C"
  will print ABC
  PRINT "FOOT("&l&")="&"BIGTOE"
  will print FOOT(l)=BIGTOE

PRINT.FORCE THING
Command
like PRINT but forces printing whether or not an
ARGUMENT list is passed to the MACRO.
Example:
  FLUBB=[
    PRINT.F "I WAS FORCED TO PRINT THIS"
    INPUT n]
  FLUBB 4
  will print "I WAS FORCED TO PRINT THIS" even
though you directly passed the value 4.

PRINT.INP THING
Esoteric Switch
stuffs and prints the input buffer with THING up
to the first NEXTLINE, if any. A subsequent INPUT
or INPUT.STR or INPUT.NAME command will accept
this information once the user has finished
editing it and has pressed the RETURN key.

PRINT.CURSOR X,Y,THING
Esoteric Switch
sets the ADM-5 cursor to the X and Y position
indicated. X varies from 0 to 79 and y from 0 to 23 with 0,0 being in the upper left corner of the
terminal screen. It forces printing like .FORCE.
"THING" is optional; without it, the cursor is
merely positioned.

PRINT.CEOL X,Y,THING
Esoteric Switch
acts just like .CURSOR but it clears to end of
line first.
**PEIORITY WRITE**

Idiosyncrasy means a COLOR 00-11, will write over another COLOR if it is greater than or equal to that COLOR.

So: $X \text{ PRIORITY } Y = \text{MAX}(X,Y)$

For example:
- $10 \text{ PRIORITY } 11 = 11$
- $01 \text{ PRIORITY } 00 = 01$

ZGRASS has two PRIORITY WRITE COLOR MODEs 16 and 17. See REVERSE PRIORITY.

```
16 17
WRITE: 00: 01: 10: 11:
00: |==|==|==|==|
10: |==|==|==|==|
01: |==|==|==|==|
11: |==|==|==|==|
```

**PR01~?T**

**THING**

Command just like PRINT but does not print the NEXTLINE at the end.

**PROMPT.FORCE THING**

Command like PROMPT but forces printing whether or not an ARGUMENT list is passed to the MACRO.

**PUNCTUATION**

Buzzword is any typed character which is not a letter, or number, or '$'. Many PUNCTUATION symbols are OPERATORS.

**PUTTAPE NUMBER,FILENAME,STRING**

Command puts FILENAME (MACRO, STRING, ARRAY, SWAP MODULE, SCREEN dump) on the tape the number of times indicated by NUMBER under the NAME of "FILENAME". The last ARGUMENT is a message to be put with the tape directory header which will print back when scanning the tape using GETTAPE. See GETTAPE. The reason for printing a file several times is to safeguard against errors. An error detection code is stored with each entry on tape and if an error is detected by GETTAPE, it will try the next copy
automatically for you. Press the RESET button to stop PUTTAPE.
Example:
PUTTAPE 3, PARMESAN, [THIS IS A SNAP OF CHEESE]
puts out the SNAP ARRAY PARMESAN three times on
tape with the message indicated.

PUTTAPE.TV
Command
a 16K dump of the screen will be put out on tape
under the FILENAME.

RADIANS
Esoteric Buzzword
PI RADIANS is defined as equal to 180 degrees.
One degree is equal to 3.14159/180 RADIANS. One
RADIUS equals 180/3.14159 degrees. (1 RADIUS =
57.296 DEGREES) SINE, COSINE, and TANGENT take
values in DEGREES. ARCTAN returns values in
DEGREES. The system default is DEGREES. If you
want to use RADIANS instead, set $RD to 1.

RANDOM
Buzzword
is a way of choosing a NUMBER in a range so that
the NUMBER is not predictable. The RANDOM
OPERATOR in ZGRASS is '%'. 10%100 means pick a
NUMBER between 10 and 100 (but not including 100).
Each time the % OPERATOR is used, the answer
should be different, because it is RANDOM,
although sometimes it's the same.

RECURSION
Buzzword
see RECURSION.

RELATIONAL OPERATOR
Buzzword
returns the value of 1 if the condition is true, 0
if false. RELATIONAL OPERATORS are used in IF
statements mostly but can be used in other
contexts as well since they are OPERATORS just
like the arithmetic ones.
The RELATIONAL OPERATORS in ZGRASS are:
OPERATOR: MEANING:
== equals
< less than
> greater than
<= or =< less than or equals
>= or => greater than or equals
# or <-> not equals
See IF COMMAND for examples.
REPLACE(BIGSTRING,OLDSTRING,NEWSTRING,NUMBER)
REPLACE(BIGSTRING,OLDSTRING,NEWSTRING,NUMBER,LOWER,UPPER)

Esoteric Swap Function

Search for the occurrence of OLDSTRING in BIGSTRING from the beginning of BIGSTRING and replace OLDSTRING with NEWSTRING. NUMBER specifies how many times to attempt replacement. The string with the replacement is returned. BIGSTRING is not modified. The matching of OLDSTRING is accomplished in the same manner as in the MATCH routine. You can use expression symbols, as described in the MATCH FUNCTION. If LOWER and UPPER are present, they indicate the start location to search and the end location. If UPPER is less than LOWER, the search is done backwards (that is, from UPPER down one by one to LOWER). Specifying a larger number to replace than is possible simply changes all the occurrences of OLDSTRING to NEWSTRING.

Examples:

```
PR REPLACE("ABA","A","-*-",1)
prints out "+_BA"
```

```
PR REPLACE("ABA","A","-*",1,5,0)
prints out "AB-*"
```

```
PR REPLACE("SUNSHINE","SUN","MOON",3)
prints out "MOONSHINE"
```

```
PR REPLACE("UNIVERSITY OF ILLINOIS AT CHICAGO CIRCLE","**","UICC",1)
prints out "UICC"
```

```
PR REPLACE("THIS IS A VERY EASY TEST",[AEIOU],["-"],20,10,0)
prints out the string "TH-S -S - VERY EASY TEST"
```

```
NOISE=\[BEEP THE JEEP\]
```

```
PR REPLACE(NOISE,"EEP","UNK",2)
prints out "BUNK THE JUNK", NOISE is unchanged.
```

```
NOISE=REPLACE(NOISE,"EEP","UNK",2)
prints out "BUNK THE JUNK", and assigns this NEWSTRING to NOISE.
```

RESOLUTION

Buzzword

is the measure of the number of PIXELS on the TV screen. The RESOLUTION of ZGRASS is 320 by 201.

RESTART

Command

clears memory and restarts ZGRASS if you answer by pressing the 'Y' key. This is a software way to push the red RST button. 'N' or any other key press will not clear memory. This option is there since system failure often results in automatic
restarts, and typing "N" in prevents you from losing everything in MEMORY.

**RESTART STRING**

Command

will RESTART the system and then automatically execute the STRING. Example:

```
RESTART [DGET DOIT;DOIT]
```

NOTE: Some differences between RESTART and RESTART STRING:

With RESTART STRING, the $VARIABLEs do not reset, the STRIPE command is still in effect, and a DLOAD'd disk is still there. Also, RESTART STRING does not ask for 'Y' or 'N'.

**RETURN**

Command

returns control to the calling MACRO. Same as running off the end of a MACRO.

**RETURN VALUE**

Command

returns the VALUE indicated and control to the calling MACRO. Useful for creating user defined FUNCTION calls which return values. Example:

```
MAX=(INPUT a,b,c;.NOTE THE local VARIABLEs
IF a<b,IF b<c,RETURN c
IF a>b,IF a>c,RETURN a
RETURN b]
```

This will return the maximum of the three parameters passed and could be used in:

```
BIGGEST=MAX(OF,THES3,THREE)
HONEY=MAX(CRUNCH1,CRUNCH2,KISS)
NUWAVE=MAX(?a,?b,?c)
```

The last is an example of passing LOCAL VARIABLEs. NOTE that, for a rather esoteric design deficiency, you cannot pass back a local string with RETURN unless the MACRO is COMPILEd or you CONCATENATE it with a null string.

**REVERSE-PRIORITY**

Buzzword

means a COLOR 00-11, will write over another COLOR if it is less than or equal to that COLOR.

So:

```
X REVERSE PRIORITY Y equals MIN(X,Y)
```

For example:

```
10 REVERSE PRIORITY 11 equals 10
```
10 REVERSE PRIORITY 01 equals 01
ZGRASS has two REVERSE PRIORITY COLOR MODES 19
(01-red), and 18 (10-green). See PRIORITY WRITE.

REVERSE 19 18
PRIORITY: 00 01 10 11
--- --- --- ---
00 00 00 00
--- --- --- ---
01 00 01 01
--- --- --- ---
10 00 10 10
--- --- --- ---
11 00 10 11
--- --- --- ---

RS232(NUMBER)
Esoteric Function
returns the INTEGER value of the RS232 PORT
indicated by NUMBER. If NUMBER=0, the terminal
port is read, if NUMBER=1, the accessory RS232
PORT is read. 0 is returned if no character is at
the PORT.
Examples:
GETAKEY= PRINT "PRESS A NUMBER KEY"
A=RS232(0)
IF A=0, SK -1
IF A>47&&A<58, PRINT "YOU PRESSED THE ",A-48," KEY"
(Note: this will not work well in .B or .F MACROS
because key presses are automatically sent to
normal (or"calculator mode").
ANYBODYSHERE=[A=RS232(1)]
IF A#0, PRINT "WAKEUP"
This will print "WAKEUP" if a device or other
computer is trying to talk to ZGRASS over the
accessory PORT. Note: since 0 indicates no
character, you cannot receive an ASCII null
character. Also note that the high bit of each
character is set to 0 automatically, unless you
set $RS to 1. Use the PORT command for more
direct control.

RS232 NUMBER1,NUMBER2
Esoteric Command
If NUMBER=0, then write to the terminal. If
NUMBER1=1, then write to the accessory RS232
PORT. NUMBER2, a VALUE from 0-255, is written to
the PORT chosen.
Example:
RS232 0,7
will make the terminal beep. A table of ASCII
values in decimal will help you with this COMMAND. 
See ASCII. Note: you can transmit an ASCII null 
character by:

RS232 1,0

RS232.GET()
RS232.GET(MATCHCHAR)
RS232.PUT STRING
RS232.PUT STRING,MATCHCHAR
RS232.SGET STRINGNAME,LENGTH
RS232.SPUT STRINGNAME,LENGTH
RS232.AGET ARRAYNAME,LENGTH
RS232.APUT ARRAYNAME,LENGTH
RS232.BGET ADDRESS,LENGTH
RS232.BPUT ADDRESS,LENGTH
RS232.RESET

Esoteric Switches
These commands and functions allow higher speed 
communication over the RS232 Auxiliary port than 
the single-byte RS232 commands. Up to 9600 baud 
(960 bytes/sec) can be transmitted or received 
using the fact that blocks or strings of bytes are 
being passed.

RS232.GET is a function that returns a string 
received over the port which is terminated by a 
null (byte value 0) or a match character (a 
NEXTLINE, perhaps) given as its decimal ASCII 
value. It and all the other GET commands will 
wait for bytes to be ready. The match character 
or null is included in the returned string.

RS232.PUT STRING sends out the string until a null 
or the optional match character specified is 
mapped. The match character or null is sent as 
well. Transmit commands do not wait for any kind 
of handshaking themselves. Any protocol must be 
established with the regular single-byte RS232 
commands first.

RS232.RESET flushes the DART chip. It has a 
three-byte buffer which may have stuff left in it 
from establishing protocols or previous 
transmissions. This switch makes sure it is empty 
and waiting for new data to be received.

RS232.BGET ADDRESS,LENGTH will accept exactly the 
number of bytes indicated by LENGTH and put them 
starting at the ADDRESS given. This is useful for 
transmitting parts of screens (the screen starts
at 16384 and has 80 bytes per line for 201 lines). Note that you must set $RS to 1 to disable auto-zeroing of bit 7 of each byte as it is received (many computers use this bit for parity so it is filtered out unless $RS=1). Receiving/sending a full 16K screen at 9600 baud takes about 17 seconds. Note that you can also receive into user RAM (32-64K) but this is quite dangerous and is the reason for .AGET, .APUT, .SGET and .SPUT.

RS232.BPUT ADDRESS,LENGTH sends the number of bytes indicated by LENGTH out starting at the ADDRESS given.

RS232.SGET STRINGNAME,LENGTH is like .BGET but it calculates the address of the first byte of the STRING associated with STRINGNAME for you and uses it. The length of the STRING must be greater or equal to the length to be received or catastrophe is likely. You can force a string to be created with a certain length with the STRING.MAKE command. Unlike with .GET, the string is not automatically created, so you must make sure there is already enough room to stuff the bytes.

RS232.SPUT STRINGNAME,LENGTH is like .BPUT but calculates the address of the first byte of the STRING associated with STRINGNAME for you. It will transmit the number of bytes given by LENGTH whether or not the end of the string is overrun. Obviously, you have to know what you are doing with these commands.

RS232.AGET ARRAYNAME,LENGTH is like .SGET but uses the first byte of the first element of the ARRAY specified. You must have dimensioned the array to have enough space to hold the number of bytes specified by LENGTH or great grief is certain (good old ERROR #3).

RS232. APUT ARRAYNAME,LENGTH is like .SPUT but uses the address of the first byte of the array to start sending bytes. .APUT and .AGET are good ways of sending SNAPs over phone lines. Remember that each element of an integer array takes two bytes and each element of a floating array takes 4 bytes.
SCALE XSCALE, YSCALE, SNAPNAME, XCENTER, YCENTER, DISPLAYMODE
SCALE XSC, YSC, SNAPNAME, Xcen, Ycen, DMODE, ROT, XDID, YDID

Command scales the SNAPNAME by XSCALE and YSCALE, and then writes it to the screen at XCENTER, YCENTER with the specified DISPLAYMODE. The range for XSCALE and YSCALE is -128.00 to 127.00. A negative scale factor will give a mirror image. SCALing by 1 on both axes will give the original SNAP. SCALing by 0 will result in ERROR #24.

Example:
Connect up JOYSTICK #1.

.LARGE
CLEAR; TEXT 0,0,2,0,1,0,0,"FROG"
SNAP RRR, 14, 5, 31, 10
X=8; Y=8
PR "PRESS TRIGGER TO SEE MORE"
IF $T1==0, SK 0
IF X>0, CLEAR; SCALE X,Y, RRR, 0, 0
IF (X=X-1)>8, Y=X, SK -3

A SNAP called RRR is made of the word FROG. This macro makes a SNAP called RRR of the screen area with the word "FROG" on it. When you pull the trigger, you see RRR scaled up by a factor of 8. When you pull the trigger again, RRR gets scaled by 7. This will continue on from 6, 5, 4, ..., -6, -7. By SCALing with a negative number, you can reverse your image.

Optionally, you can include rotation by specifying 0-3 for ROT. Also optional are XDID and YDID which are fudge or diddle factors applied to make the pixel detail loss accompanying any scale down less objectional. They essentially offset the sampling counters by the amounts specified and thus affect the aliasing somewhat.

SCALE.SCREEN XSCALE, YSCALE, 0-15, XCENTER, YCENTER, DISPLAYMODE
SCALE.SCREEN XSC, YSC, 0-15, Xcen, Ycen, DMODE, ROT, XDID, YDID

Command same as SCALE but uses contents of screen 0-15 instead of a SNAP name.

SCALE.PAN XSCALE, YSCALE, PANNAME, XCENTER, YCENTER, DISPLAYMODE
SCALE.PAN XSC, YSC, PANNAME, Xcen, Ycen, DMODE, ROT, XDID, YDID

Command same as SCALE but uses the "super snap" given by PANNAME.
SCREEN
Idiosyncrasy

UV-1's have 16 screens of 16K bytes (320 X 201 PIXELS) each. These are known as SCREENS 0-15. $TV is set to 0 on start-up and when changed, a different 16K screen is shown on the television. $MW controls which screen the computer writes to (and reads from in the case of non-PLOP color and display modes), so you can be building an image on one screen while seeing another. If $ML (memory lock) is set to 1, $MR is used for reads and $MW for writes, thus allowing more complex screen writes.

CTRL+B resets $TV, $MW, $ML, and $MR to zero, as does RESTART.

If a disk is DLOAD'd, $MW and $MR are used modulo 4, since DLOAD uses screens 4-15. For example:

```
DRAWSCREENS=[NUM=0;A=5
 1BEG $TV=NUM;$MW=NUM;CLEAR
  CIR 0,0,A,1,20;CIR 0,0,A+10,1,20;
  CIR 0,0,A+20,1,20
  A=A+10
  IF (NUM=NUM+1)#16,GOTO 1BEG]

CYCLE=[LIM=15;N=0;S=1
 1AGAIN $TV=N
  IF (N=N+S)#LIM,GOTO 1AGAIN
  S=-S
  IF LIM=15,LIM=0;GOTO 1AGAIN
  LIM=15;GOTO 1AGAIN]
```

DRAWSCREENS uses $MW to write to each of the 16 screens. Setting $TV allows you to watch the screens on the TV screen as they are being drawn on. This step is optional. CYCLE uses $TV to flip through the screens.

Note that $TV is used mod 16 by the system but can hold values over 15. $MW and $MR, however, are read and converted to mod 16 once accessed by the system so they do not exceed 15 in value under normal circumstances.
SCROLL Xcen,Ycen,Xsize,Ysize,Xmov,Ymov,Displaymode,Fcolor

Command
moves an area of the screen centered at Xcen,Ycen of Xsize, Ysize dimensions with Displaymode in the direction defined by Xmov,Ymov using Fcolor to fill in the old area. This move is performed within a window defined by Xcen, Ycen, Xsize, Ysize. Fcolor can be any one of the 4 Colors 0-3.

For example:
SIDEWAYS=[XMOVE=1;YMOVE=1
TEXT -100,-50,3,3,1,0,1,"HELLO"
TEXT -50,0,3,3,2,0,1,"HELLO"
TEXT 0,50,3,3,3,0,1,"HELLO"
AGAIN SCROLL 0,0,320,200,XMOVE,YMOVE,0,0
IF (XMOVE=XMOVE+1)<24,GOTO AGAIN]

SEMANTICS
Buzzword
The meaning of a COMMAND as opposed to its SYNTAX.

SHOW Fontarray, Character, Yoffset, Xleft, Xright, Xsize, Ysize
Swap Command
puts the information concerning the character in the FONT ARRAY specified in the variables indicated. See FONT.

SHRINK Xscale,Yscale, Name, Xcenter,Ycenter, Xsize,Ysize
Command
is like SNAP but shrinks or expands the part of the screen it is SNAPping. Only positive VALUES for Xscale and Yscale will work. Xsize and Ysize are the dimensions of the area to be shrunk.
Example:
BOX 0,0,320,201,1
BOX 50,0,40,201,2
BOX 0,50,320,60,6
SHRINK .25,.3,SCREEN1,0,0,320,201
CLEAR
DISP SCREEN1,0,0,0
SHRINK .25,.3,SCREEN2,0,0,320,201
DISPLAY SCREEN2,0,0,0
CLEAR
DISP SCREEN1,-50,0,0
DISP SCREEN2,50,0,0
SINE(NUMBER)
Function
returns the sine of NUMBER.

SKIP NUMBER
Command
skips the given NUMBER of lines (excluding the one you are on). It transfers control by counting the NUMBER of NEXTLINE's indicated. SKIP 0 hangs in place, SKIP 2 skips the next 2 lines, SKIP -3 goes back 3 lines. SKIP 999 is the same as RETURN and SKIP -999 will get you back to the beginning of the MACRO. SKIP does not allow LABELs. Use GOTO with LABELs.
Examples:
SKIP 0; .GOES TO THE BEGINNING OF THIS LINE
SKIP 2; .SKIPS THE NEXT TWO LINES
SKIP -3; .GOES BACK 3 LINES
SKIP 1; .GOES TO THE NEXT LINE

TOGO=[m=10
PRINT m,"TO GO"
IF (m=m-1)<0,SKIP -1
PRINT "NO MORE"]

SNAP NAME, XCENTER, YCENTER, XSIZE, YSIZE
Command
takes the PIXELs in the area indicated and saves them in an ARRAY called NAME. The DISPLAY COMMAND is used to redraw the ARRAY somewhere else. The SCALE COMMAND is used to scale and redraw the ARRAY somewhere else. NAME(0) gets the XSIZE and NAME(1) gets the YSIZE for your use. Example:
FLASH=[s=28
BOX 0,0,s,s,5%8;IF (s=s-2)>2,SK 0
SNAP ART,0,0,32,32
DISP ART,x=x+$X1,y=y+$Y1,0;SKIP 0]
FLASH
FLASH will draw some BOXes, make a 32X32 SNAP called ART, and finally allow the user to move the SNAP around on the screen using JOYSTICK #1.

NOTE: The largest square area you can SNAP in one piece is 125X125 PIXELs.

Another note: if you do a snap that extends over the edge of the screen (for example, SNAP JERQ,150,95,40,40) Zgrass will subtract the amount over the edge on each axis from the dimensions specified and recenter the newly-figured SNAP as...
if you snapped the same visible area without exceeding the screen edge(s). You can avoid confusion by not snapping stuff partially off-screen, of course. (or about 15625 PIXELs or 1/4 of the screen).

SNAPPED PIX
Buzzword
is a special ARRAY which contains PIXELs from an area on the screen specified by a SNAP COMMAND. See SNAP and DISPLAY.

SORT(NUMBER)
Function
returns the square root of NUMBER.

STATUS XCENTER,YCENTER
Command
returns the X,Y COORDINATEs of the current center of the screen in XCENTER,YCENTER. See WINDOW.CENTER.

STATUS LEFTX,RIGHTX,TOPY,BOTTOMY
Command
returns two X COORDINATEs and two Y COORDINATEs which describe the boundaries of the current WINDOW. See WINDOW.

For those of you who noticed, Zgrass has an internal ANYARGS command that lets it know how many arguments have been passed, which is how STATUS knows when to give you WINDOW coordinates and when to give you CENTER coordinates.

STOP NAME
Command
is used to selectively halt the EXECUTION of a MACRO or COMPILED MACRO running in .B or .F mode. A MACRO/COMPILED MACRO can stop itself or any other MACRO.

STRING
Buzzword
is a collection of characters (numbers, letters, punctuation) delimited (enclosed) by single or double quotes or balanced square '[]' or curly '{}' brackets. If you have to use a string delimiter within a STRING, make sure it is delimited by a different string delimiter or things will get very confused (most likely, it will consider the rest of your MACRO as part of
the STRING). Examples:
"THIS IS A STRING"
"PRINT A*B*C
SKIP -1; THIS STRING COULD BE A MACRO TOO"
[1234]
PRINT [' ]; A QUOTE IN A STRING

STRING(NAME, NUMBER)
Esoteric Function
returns the INTEGER which represents the character in the position indicated by NUMBER. Can be used to access STRINGS as BYTE ARRAYS.
Example:

```
TYPE=
PRINT "INPUT A STRING OF CHARACTERS"
INPUT. STR CHAR
A=0
B=STRING(CHAR, A)
IF B==0, SK -4
PRINT B," IS ASCII FOR", ASCII(B); A=A+1; SK -2
SK -6]
```

This prints out the decimal ASCII values of the string of characters which you input and are stored in CHAR. If you input "ABC", you should get 65, 66, 67. The listing of characters stops when it encounters the null (INTEGER value 0) at the end of CHAR (and every STRING).

STRING NAME, NUMBER1, NUMBER2
STRING NAME, NUMBER1, NUMBER2, ..., NUMBERn
Esoteric Command
puts NUMBER2 into the STRING "NAME" offset by the number of BYTES in NUMBER1. If more bytes are indicated by arguments following NUMBER2, they are stuffed in sequential bytes.
Example:

```
LETTERS="ABCDEF"
STRING LETTERS, 3, 50, 52
PRINT LETTERS
```

will print ABC24F

Note: allowing NUMBER1 to exceed the length of the STRING can clobber innocent MEMORY and lead to software failures. You can use a STRING as a BYTE ARRAY only if you have first made it large enough by CONCATENATION, ASSIGNMENT or STRING.MAKE. This command and the ASCII command are potentially useful for communication over the accessory RS232 PORT.
STRING MAKE NAME, LENGTH

Esoteric Switch
allows you to easily create a string for use as a byte array. The bytes are not cleared, nor is there a null at the end, so make sure you put meaningful stuff there before you PRINT or otherwise reference the string.

STRING VARIABLE

Buzzword
is a NAME that has a STRING as its VALUE.

STRIPE STRIPENUM, LINENUM, COL0, COL1, COL2, COL3
STRIPE.OFF
STRIPE.STR STRINGNAME

Esoteric Command
used to change the left COLORMAP part way down on the screen. The STRIPENUM (range 0-15) is an index into a special 80 byte system table of 5 byte entries. The LINENUM can range from 0 to 196. It indicates how far down the screen the change should start. (50 is 1/4 down, 100 halfway, 160 is 4/5 down, etc.) The LINENUM indicates approximately where COL0 gets changed. COL1 gets changed the next video line, COL2 the next, and COL3 the next (the hardware does not support this useful function well and leaves only 11 microseconds to change each element during a scanline). COL 0-3 are numbers in the range 0-255, representing the respective colors the pixel values for what is normally $L0, $L1, $L2 and $L3 on the screen in that particular stripe. You must leave at least seven lines between stripes. Furthermore, unless you want the screen to flash, make sure the LINENUMs get larger as the STRIPENUM gets larger. You can cancel STRIPE by STRIPE.OFF. Note that it is normal for the stripes to temporarily undo during disk access or when using the TABLET, BREAK, AND CTRL+C. The first stripe should look like:

STRIPE 0, 1, COL0, COL1, COL2, COL3

where COL0-3 are the numerical values of the colors you want. The fact that you are specifying line 1 for the 0th stripe to start at is important. If you specify line 0, the system will assume no stripes are enabled.

The 80-byte system table can be loaded by a byte array cleverly set by you to be the last five arguments of the STRIPE command that would be necessary to achieve the same effect. (The first
argument is not used because the position in the array assumes which stripe it is.) STRIPE.STR
STRINGNAME can then be used to tell the system to load the byte array specified by STRINGNAME into the 80-byte system table at the next vertical interrupt. Thus, you can switch stripes quickly and cleanly. Be careful, as usual, when storing byte arrays as strings—since strings are terminated by nulls (which have the value 0), you cannot store a byte array with 0's in it unless you fudge it somehow.

NOTE:

RESTART STRING
will clear memory without changing stripes, although the screen will flash momentarily.

Use STRIPE.OFF to clear STRIPEs from the screen.

SUBSTR(MYSTRING,BEGIN,END)
Esoteric Function
returns a STRING value that is the subset of MYSTRING specified by the BEGIN and END displacement values. If the END value extends beyond the end of MYSTRING, the substring simply contains all the characters of MYSTRING following BEGIN. A null string is returned if the value of BEGIN extends beyond the end of MYSTRING.

Examples:
PR SUBSTR("ABCDEF",0,2) prints out the string "ABC"
PR SUBSTR("ABCDEF",4,20) prints out the string "EF"

SWAP COMMAND or FUNCTION
Idiosyncrasy
is a COMMAND or FUNCTION written in assembly language which must first be gotten into memory from disk or tape.

SWITCH
Buzzword
is an option for COMMANDS, FUNCTIONS, and MACROS. The only switches defined for MACROS are .B and .F which cause the MACRO to be EXECUTED in the background and foreground respectively. Many COMMANDS and FUNCTIONS (INPUT, ARRAY, etc.) have SWITCHes which are given as separate entries in this glossary. SWITCHes are always preceded by the NAME they are modifying and a period.

Examples:
INPUT.STR SAM
ARRAY.INT FOO,123
DEATHWEAPON.B

SYNTAX
Buzzword
is the form of a language, its spelling, punctuation, words, etc. (Contrast with SEMANTICS.)

TABLET(X,Y)
Function
returns the X,Y values of the TABLET pen position in X and Y, and the value of the pen push (0=not pushed but on surface, 1=pushed, -1=off surface). If you have a four-button cursor, the value returned also indicates which button was pushed. Example:

PXY=[A=TABLET(X,Y)
X=X/6;Y=Y/6
IF A==1,BOX X,Y,4,4,3
IF A==0,BOX X,Y,4,4,5;BOX X,Y,4,4,5
SKIP -4]

This will put a blue BOX (if the pen or yellow button on the cursor is pushed) or a flashing red BOX at the pen's or cursor's current location. The X and Y range is:

-1100 < X,Y < 1100

Divide X and Y by 6 to SCALE them to Zgrass X and Y coordinate range. You can also use this scaling factor to manipulate the relationship between the tablet area and screen memory. For example, when tracing a small object, divide by a smaller number so the image fills more of the screen.

Of course, any VARIABLE NAME can be used instead of X and Y. NOTE that if TABLET returns a -1, you should not rely on the values of X and Y.

TANGENT(NUMBER)
Function
returns the tangent of NUMBER.

TERMINAL
Esoteric Command
TERMINAL bypasses the keyboard and puts the CRT directly in connection with the accessory RS232 PORT so you can connect up to another computer system as a terminal. BREAK gets you back to ZGRASS.
TERMINAL ARGO, ARG1, ... ARG9

Esoteric Command
allows user to specify one of two terminals with ARGO. Set ARGO to 0 for Hazeltines, and 1 for ADM-5's. Then, up to 9 decimal ARGUMENTS may be entered. ARG1 allows you to define an additional key for rubout outside EDIT (ESC or underscore work well). ARG2-9 specify the EDIT keys:

Maps into:

ARG2  CURSOR Right  08 (^H, Backspace)
ARG3  CURSOR down   09 (^I, Tab)
ARG4  CURSOR Up      010 (^J, Linefeed)
ARG5  CURSOR Left    011 (^K)
ARG6  INSERT char    94 (^)
ARG7  Delete Char    18 (^R, HOME)
ARG8  Delete Line    01 (^A, CLEAR)
ARG9  Extra for RUB  127 (RUBOUT)

Examples:
SETUP=[TERMINAL 0,95,8,11,26,24,94,9,27,95]
SETS

Sets up for an Hazeltine using underscore as an alternative for DEL (rubout) both in and out of EDIT. It also specifies the arrow keys for cursor left, right, up, and down in EDIT. Delete character, in this example, is TAB and delete line is ESC. You need an ASCII table for your terminal to use this command successfully.

ADM5=[TERMINAL 1,95,12,10,11,8,30,27,9,95]
ADM5

Sets up for an ADM5.

TEXT XLEFT, YLOWER, HORSP, VERSP, FCOLOR, BCOLOR, DMODE, TSTRING, FONTARRAY1 ... FONTARRAYn

Command is used to generate strings of text or arbitrary figures on the TV screen. The size of the text, the styles, the colors, and spacing are all user-definable through the FONT COMMAND and the TEXT COMMAND itself.

ARGUMENT: Description:

XLEFT is the X COORDINATE where TSTRING is to begin.
YLOWER is the bottom row of pixels on which TSTRING is to be displayed.

HORSP is any positive or negative integer or zero. It represents a constant spacing factor in pixels to be inserted between characters.

VERSP is an integer which signifies the number of pixels to move up (+) or down (-) on seeing a NEXTLINE in TSTRING.

FCOLOR is the foreground color of the character (0-3).

BCOLOR is the background color of the character (0-3).

DMODE is the DISPLAY MODE. Any ZGRASS DISPLAY MODE can be used.

TSTRING is the STRING to be displayed. Every character in the STRING should have been previously defined in a Font Array named in the next operand. If a character isn't found in one of the named Arrays, the character is ignored, and no warning is given.

FONTARRAY1, FONTARRAYn are the Names of the Font Arrays to be used. The Arrays are searched in the order given. The number of Arrays that can be entered is only limited by the number of characters you can type on a line. The default FONTARRAY is used if none is specified.

For example:

WRITEIT=
X=-100;Y=50
TEXT X,Y,3,4,1,0,0,"THIS IS A TEXT"
TEXT X,Y-20,3,4,2,0,0,"WITH DIFFERENT COLORS"
TEXT X,Y-40,6,4,3,0,0,"AND VARIABLE SPACING"

This example uses the default FONTARRAY.

Note that you can use the TEXT command as a function as well, in which case it returns the x-coordinate of the first pixel following the last pixel of the last character drawn. This is the probable place to start the next TEXT command's X. It is also useful for calculating the x-width in
pixels of the entire string of characters as drawn which can be used for centering lines, for example.

TEXT.ROT 0-3, plus same arguments as TEXT
Command

0-3 specifies the rotation of the text; 0 = no rotation; 1 = 90 degrees; 2 = 180 degrees; 3 = 270 degrees. For example:

\[
\text{ROTATETEXT} = \text{TEXT.ROT 1,-100,-50,3,3,1,0,0, "TURN YOUR HEAD"}
\]

\[
\text{TEXT.ROT 2,20,0,3,3,2,0,0,"AROUND"}
\]

\[
\text{TEXT.ROT 3,100,60,3,3,3,0,0,"TO READ THIS!"}
\]

TEXT.SPACE SPACEARRAY, plus same arguments as TEXT
Command

SPACEARRAY, a text-spacing array described in FONT, is used to affect the spacing of characters.

TEXT.SPROT 0-3, SPACEARRAY, plus same arguments as TEXT
Command

does both .SPACE and .ROT.

TIMEOUT NUMBER
Esoteric Command

wait for NUMBER/60 seconds and then return.

Example:

\[
\text{FOO} = \text{[TIMEOUT 300}
\]

\[
\text{PRINT "5 SECONDS UP"]
\]

\[
\text{FOO.F}
\]

Every 5 seconds "5 SECONDS UP" will be printed. Works only with .F macros.

TRUTH TABLES
Buzzword

See AND, OR, PLOP, PRIORITY, REVERSE-PRIORITY, and XOR.

TXT X,Y,XSIZE,YSIZE,FCOLOR,BCOLOR,DISPLAYMODE,CHARSTRING
Swap Command

prints CHARSTRING on the TV screen starting at X,Y with the character size specified by XSIZE,YSIZE, in FCOLOR with BCOLOR as the background COLOR using the specified DISPLAYMODE. The smallest values for XSIZE,YSIZE are 1,1 which means that the characters will be 5 PIXELs wide and 7 PIXELs high. The largest character can take up 4K or the largest available chunk of memory.

Examples:

\[
\text{TXT 0,0,1,1,1,0,0,"SMALLTEXT"}
\]

this will print "SMALLTEXT" starting at 0,0 with
5X7 characters in red (01) with a white background (00) using the PLOP DISPLAYMODE.

TXT -50,-30,2,2,3,1,1,"SMALLTEXT * 2"
will print "SMALLTEXT * 2" starting at -50,-30
with 10X14 characters in blue (11) with a red
background (01) using the XOR DISPLAYMODE.

USEMAP Command
gives a list of NAMES currently in use and the
number of BYTES they take up.

VALUE Buzzword
is typically a NUMBER or STRING. PRINT will
always tell you the value of a CONSTANT or a
VARIABLE.

VARIABLE Buzzword
is a NAME you can use to hold a VALUE. Any NAME
in ZGRASS that can be put on the left side of a
'=' is a VARIABLE and its VALUE can be varied by
that ASSIGNMENT (which is why it's called a
VARIABLE instead of a CONSTANT, of course).
USEMAP will give you information about your
VARIABLEs. NOTE: VARIABLEs A-Z and the DEVICE
VARIABLEs are built into the system and are not
listed in USEMAP.

VERSION Function
returns the VERSION number of the current ZGRASS
software you have. Example:

PRINT VERSION()
should print -6.

WAIT NUMBER Command
waits the specified NUMBER of seconds before
continuing by doing a SKIP 0 until the time is up.
WAIT only works with whole seconds; fractions have
no effect. Also note that WAIT will not compile.
Example:

NEST=[A=0
A=A+10
BOX 0,0,A,A,7
WAIT 2
IF A<200,SK -3]
This will draw a BOX waiting approximately 2
seconds before starting another. To wait a
fraction of a second, use a System Timer which counts in 1/60 seconds.

TENPERSECOND=[$ZO=6
IF $ZO#0, SK 0
PR "XX"
SK -3]
this will print "XX" ten times per second if compiled first.

WHATSIS(NAME)
Esoteric Swap Function
returns an INTEGER value for the type represented by the NAME.

Values: Meaning:
2  ;Null NAME
8  ;STRING NAME
14 ;NUMBER NAME
16 ;ARRAY NAME
18 ;COMPILED MACRO NAME
20 ;SWAP MODULE

Example:
MACROONLY=[GETTAPE SUE
A=WHATSIS(SUE)
IF A#8, SK -2]
This will set A to SUE's type. If you PUTTAPEd a SUE that was a SNAP and a SUE that was a MACRO, waiting for A to equal 8 would allow you to skip the SNAP called SUE.

WINDOW XLEFT,YBOTTOM,XRIGHT,YTOP
Command
creates a window in the ZGRASS screen with XLEFT as the left side, YBOTTOM as the bottom side, etc. CLIPPING is done for all drawing COMMANDS. Windows are CLIPPED to the screen and use the same COORDINATE system unless changed by CENTER. WINDOW FULL resets the WINDOW to full screen. (Screen dumps are not subject to the WINDOW command.)

Example:
CLEAR
WINDOW -40,-60,40,60
VIEW=[BOX -160%159,-100%100,20,20,5%8
SKIP -1]
Note that CLEAR.WIND clears the window area. Also note that screen dumps (DPUT.TV's) are not subject to windows. If you must get a screen dump into a window, get it onto a screen you are not using and then use DISPLAY.SCREEN which does obey window
boundaries.

WINDOW.BOX XCT, YCT, XSIZE, YSIZE
Command

is the same as WINDOW except you specify it like
BOX using XCT, YCT to mark the center and
XSIZE, YSIZE to specify the dimensions of the
WINDOW.

WINDOW.CENTER XCOOR, YCOOR
Command

changes the center of the screen, default of which
is 0,0. See STATUS.
Example:
BOX 10,10,20,20,1
WINDOW.CENTER 160,100
BOX 10,10,20,20,1
WINDOW.CENTER will change the center of the screen
to the lower left corner.
This could be useful for roaming around large
databases like the map of a city. Use STATUS to
find the current screen WINDOW.CENTER.

WINDOW.FULL
Command

resets the WINDOW to full screen.

WRAP XCN, YCN, XSIZE, YSIZE, XMOVE, YMOVE, DISPLAYMODE
Swap Command

moves an area of the screen centered at XCN, YCN
of XSIZE, YSIZE dimensions in the direction defined
by XMOVE, YMOVE around onto the originally defined
area by wrapping around using the specified
DISPLAYMODE. For example:
MOVEIT=[
A=0; B=10
BOX 0,0,0,B,20; IF (B=B+10)<100, SK 0
1BEG WRAP 0,0,320,200,A*2,-A*2,0
IF (A=A+1)#25, GOTO 1BEG]

/WRAP AROUND
Buzzword

is the phenomena that causes OVERFLOWed VARIABLEs
to print as weird numbers. If a DEVICE VARIABLE
OVERFLOWS at 255, 256 WRAPS AROUND to 0, 257 to 1,
etc. This is the same as modulus arithmetic with
base 256.
XOR

Buzzword
(also called 'exclusive or') is a LOGICAL operation used to draw PIXELs on the screen. What gets drawn is a value from 0-3 and is computed by the XOR function of what there was on the screen with what you give it to write there. The reason for this complexity is that a couple of neat tricks are made possible by XOR. First, if you draw anything on the screen with XOR (COLOR MODES 4-7) or DISPLAY a SNAPped picture element with DISPLAY MODE 1, you can erase it by simply drawing or DISPLAYing it again the same way. In other words, two XOR's is the same as nothing. Second, by setting SL3=SL2 (and SR3=SR2 if you mess with $HB), you can make anything written with COLOR 1 pass 'behind' anything written with COLOR 2 (you have to try it to believe it).

XOR table using 2 BITS:

<table>
<thead>
<tr>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOR</td>
<td>00</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>===</td>
<td>===</td>
<td>===</td>
<td>===</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>===</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>00</td>
<td>11</td>
</tr>
<tr>
<td>===</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>11</td>
<td>00</td>
</tr>
<tr>
<td>===</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>10</td>
<td>01</td>
</tr>
<tr>
<td>===</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

The XOR COLOR MODES are 4-7. The XOR DISPLAY MODES are 1,11,21,...,131,141.

ZAP1(ADDRESS)
Esoteric Swap Function
returns the 8-bit byte at the ADDRESS given.
Example:

DUMP=[A=0
PRINT ZAP1(A)
A=A+1
IF A<32767,SKIP -2]
This will print a decimal dump of the ZGRASS code for anyone who is into machine code disassembling.
ZAP1 ADDRESS, VALUE
Esoteric Swap Command
puts the 8-bit number given by VALUE into the ADDRESS given. You can use this command to plop 4-pixel groups onto the screen if the ADDRESS is between 16384 and 32767. Be careful, this command can wipe out the system if you use negative addresses that is, in user RAM.

ZAP2 (ADDRESS)
Esoteric Swap Function
returns the 16-bit value at ADDRESS.
Example:
   PRINT ZAP2(1)
prints out the first location the code jumps to on RESTART.

ZAP2 ADDRESS, VALUE
Esoteric Swap Command
puts the 16-bit number indicated by VALUE into the two bytes starting at ADDRESS.