DEVICE CONVENTIONS IN GRASS2

"DEV" IN GRASS GENERALLY MEANS ANYTHING KNOWN TO THE SYSTEM THAT CAN HOLD A VALUE FROM -32768 TO +32767 (THE LIMITS OF A 16-BIT NUMBER). BY CONTRAST, FLOATING POINT NUMBERS IN GRASS HAVE MUCH HIGHER RANGES AND TAKE UP MORE BITS AND ARE RELATIVELY TIME-CONSUMING TO CONVERT TO INTEGER. 16-BIT NUMBERS ARE WHAT THE VECTOR GENERAL (VG) FEEDS ON, AND FEEDING THE VG IS WHAT GRASS IS ALL ABOUT AT MACHINE LEVEL.

SO, THERE'S A BASIC DICHOTOMY IN GRASS ARITHMETIC BETWEEN FIXED AND FLOATING POINT NUMBERS. ALL DEV'S ARE FIXED POINT, AND CAN BE TIED DIRECTLY TO THE 16-BIT REGISTERS IN THE VG. DEV'S CURRENTLY INCLUDE DIALS, FIXED VARIABLES, FIXED ARRAYS AND, IN SOME CASES, FIXED POINT EXPRESSIONS. THUS, IN

\[ A = B + \frac{D}{I0} \]

EVERYTHING BUT THE 10 AND THE \( =.+/ \) ARE DEV'S BUT IN

\[ FA = K \sin(FB) * FC \]

ONLY THE \( K \) IS A DEV (SEE HELP HELP FOR DEFINITIONS HERE).

NOW, GRASS USES ARITHMETIC VARIABLES THAT ALREADY HAVE NAMES. THIS ELIMINATES THE DECLARE STATEMENTS THAT ARE SO CUMBERSOME IN MANY PROGRAMMING LANGUAGES AND ALSO VASTLY SIMPLIFIES WHAT GRASS HAS TO DO TO FIGURE OUT WHAT YOU'VE TYPED. SOME-DAY, WE MAY DO SOMETHING ABOUT LETTING PEOPLE HAVE ARBITRARY VARIABLE NAMES, BUT FOR THE TIME BEING, THE LEGAL DEV'S ARE:

ARITHMETIC:
- \( A-Z \)
- \( VA-VZ \)
- \( WA-WZ \)

LOCAL ARITHMETIC:
- \( LA-LZ \)

ARRAY:
- \( AA(\ldots)-AMC(\ldots) \)

SPECIAL ARITHMETIC:
- \( O A-O H \) NORMAL EXCEPT ALSO PUT OUT VOLTAGES TO THE IMAGE PROCESSOR. 0=0 VOLTS, 127=+1 VOLT, -128=-1 VOLT (EXPLAINED MORE LATER).
- \( SA-S H \) CONVERSION TO SINE FUNCTION (EXPLAINED LATER)
- \( CA-C H \) CONVERSION TO COSINE FUNCTION (EXPLAINED LATER)
- \( MA-MZ \) AND \( NA-NZ \) LINEAR TIME-BASED VARIABLES (SEE BELOW FOR DETAILED EXPLANATION)
- \( QA-QZ \) AND \( RA-RZ \) SINUSOIDAL TIME-BASED VARIABLES

INTERACTIVE CONTROL DIALS, ETC:
- \( DO-D9 \) TEN CONTROL DIALS
- \( SO-S9 \) TEN SLIDE POTENTIOMETERS
- \( PO-P3 \) FOUR MORE CONTROL DIALS
- \( JX, JY, JZ \) JOYSTICK #1
- \( KX, KY, KZ \) JOYSTICK #2
- \( TX, TY, TZ \) TABLET (TZ=1 FOR PRESS, 0 FOR NEAR & -1 FOR FAR)

DEV'S CAN BE USED IN ARITHMETIC EXPRESSIONS (EXCEPT THAT YOU CANNOT WRITE TO DIALS/POTS/Etc.) AND THEY ALSO CAN BE ATTACHED TO ROTATIONS, TRANSLATIONS, SCALING AND SO ON. THIS LATTER CONVENIENCE ACTUALLY REQUIRES SOME COMPLICATED SYSTEMS PROGRAMMING, WHICH YOU SHOULDN'T WORRY ABOUT. ANYWAY, WHEN BEING USED IN RO-TATE, MOVE, SCALE, CUTOFF, SETINT, SETCO, AND SETORG, THE FOLLOWING RULES APPLY TO DEV'S:

RULE 1: IF THE DEV IS SPECIFIED ALONE, THE SYSTEM LOOKS UP ITS VALUE AND USES IT EVERY 1/60 SECOND. THIS IS WHAT GIVES THE INSTANT-SEEMING RESPONSE TO CHANGING DIALS, VARIABLES, Etc.

EXAMPLE: SCALE PIX, DO

RULE 2: IF THE DEV IS PART OF AN EXPRESSION, IT IS EVALUATED ONE-TIME ONLY AND USED AS A CONSTANT.

EXAMPLE: SCALE PIX, DO/100+A

PIX IS SCALED ON THE VALUE OF DO/100+A AT THE INSTANT THE SCALE COMMAND IS EXECUTED. IF A OR NO CHANGE, THE SCALE IS
RULE 1: IF AN = SIGN PRECEDES A FIXED-POINT EXPRESSION, THE EXPRESSION IS AUTOMATICALLY COMPILED AND THEN RE-EVALUATED EVERY 1/60 SECOND. THIS TAKES A SMALL AMOUNT OF MEMORY SO IT SHOULDN'T BE USED FOR SILLY THINGS LIKE SCALE SAM.=0 WHICH IS COVERED BY RULE 1 ABOVE. EXAMPLES: SCALE BEAN.==DO/20 SCALE BEEP.=A+C*(D-E) MOVE BLOOP.=CA+SA.=CB+SB.=CC+SC

RULE 4: IF A DEV OBEYING RULE 1 IS THE ONLY ONE SPECIFIED IN A COMMAND EXPECTING MULTI-DEV'S, THE DEV'S LOGICALLY FOLLOWING THE ONE SPECIFIED ARE USED AUTOMATICALLY. FOR EXAMPLE: MOVE SAM,DO USES DO FOR X TRANSLATION, D1 FOR Y TRANSLATION AND D2 FOR Z TRANSLATION. ROTATE SAM,Y,DO,D1,VL USES VL FOR X ROTATION ORIGIN, VM FOR Y ROTATION ORIGIN AND VN FOR Z ROTATION ORIGIN. SIMILARLY, THE CUTOFF COMMAND EXPECTS SIX DEV'S SO CUT SAM, DO3 USES D3 FOR X HIGH BOUNDARY, D4 FOR X LOW BOUNDARY, D5 FOR Y HIGH, D6 FOR Y LOW, D7 FOR Z HIGH AND D8 FOR Z LOW. SEE THE HELPS FOR EACH COMMAND TO SEE HOW MANY DEV'S ARE EXPECTED.

NOTE: WHEN RULE 2 IS FOLLOWED IN SPECIFYING A DEV (INCLUDING CONSTANTS), THE PICTURE DOES NOT NEED TO BE FIX'ED

RULE 5: IF DEV'S NOT OBEYING RULE 1 ARE USED, DEV'S MISSING AT THE END ARE ASSUMED TO TAKE DEFAULT VALUES (0 FOR MOVE, SETORG, SETINT, SETCO, +32767 FOR SCALE (ALL SWITCHES) AND CUTOFF X,Y,Z HIGH VALUES, AND -32768 FOR CUTOFF X,Y,Z LOW VALUES). EXAMPLES: MOVE SAM,1000 IS THE SAME AS MOVE SAM,1000.0,0 AND MOVE SAM,DO,1000 IS THE SAME AS MOVE SAM,DO,1000.0 AND MOVE SAM,DO,D1 SETS Z TO ZERO AND MOVE SAM,=DO/10 LEAVES Y AND Z SET TO ZERO

RULE 6: IF THE ARGUMENTS ARE NULL (I.E. "" ) THE DEV SETTING IS LEFT ALONE. FOR EXAMPLE: MOVE SAM,1000 MOVES SAM TO 1000,0,0 FOLLOWED BY MOVE SAM,,-1000 MOVES SAM TO 1000,-1000,0 THIS IS PARTICULARLY USEFUL IN THE CUTOFF COMMAND BECAUSE THE DEFAULT VALUES ARE SETUP AS: CUT SAM,32767,-32768,32767,-32768,32767,-32768 WHICH IS NOT SO MUCH FUN TO TYPE IF YOU ONLY WANT TO CHANGE THE Z-HIGH COMPONENT.

NOTE: WHEN RULE 2 IS FOLLOWED IN SPECIFYING A DEV (INCLUDING CONSTANTS), THE PICTURE DOES NOT NEED TO BE FIX'ED

SPECIAL DEV'S

1. THE DIALS, POTS, ETC. CANNOT BE WRITTEN INTO, THAT IS, THEY CANNOT BE ON THE LEFT SIDE OF AN EQUAL'S SIGN OR USED IN GETPOINTS

2. OA-CH ARE DIGITAL-TO-ANALOG OUTPUT VARIABLES THAT MAY BE WRITTEN INTO AND READ FROM AS NORMAL EXCEPT THAT THEY ALSO PRODUCE VOLTAGES IN A BOX BY THE IMAGE PROCESSOR. THE RANGE IS 1 VOLT =127, -1 VOLT =-128, AN 8-BIT RANGE.

3. SA-SH AND CA-CH ARE SPECIAL BECAUSE THEY CAUSE A SINE/COSINE CONVERSION WHEN ASSIGNED (I.E. APPEAR ON THE LEFT SIDE OF THE EQUALS SIGN). FOR EXAMPLE: SA=0 (SEE HELP FORTRAN) PROM SA GIVES 0 (SEE HELP PROMPT) SA=256 PROM SA GIVES 32767

180 DEGREES = PI RADIAN = 512 FOR SA-SH AND CA-CH. SC EACH CHANGE OF 1 EQUALS A CHANGE OF APPROXIMATELY 1/3 DEGREE. THESE VARIABLES ARE USEFUL FOR BINARY MAKING SINUSOIDAL MOTIONS. NOTE THAT THE CONVERSIONS WILL NOT TAKE PLACE IN GETP SAM,10,SA,SB,SC,SD. THE CONVERSIONS ONLY TAKE PLACE "OVER" EQUAL SIGNS
5. TIME BASED VARIABLES


BASICALLY, THE IDEA IS TO SET THE TIME VARIABLE AND SET THE CORRESPONDING GOAL VARIABLE (OA CORRESPONDS TO RA, MJ TO NJ, ETC.) ON THE SAME LINE. IT IS IMPORTANT TO SET THE TIME VARIABLE FIRST, BECAUSE THE SYSTEM WILL IMMEDIATELY UPDATE THE CORRESPONDING GOAL VARIABLE TO ITS FULL VALUE IF THE TIME VARIABLE IS ZERO (WHICH IT PROBABLY WOULD BE IF YOU SET THE GOAL VARIABLE FIRST). IF YOU DON'T TOUCH THE M'S AND Q'S, YOU CAN USE THE R'S AND N'S AS NORMAL VARIABLES.

SOME EXAMPLES:

MOVE SAM, NA
MA=0; NA=-1000; MA=180; NA=10000

THIS WILL TAKE THREE SECONDS TO GO FROM -1000 TO 10000

SCALE SAM, RC
QC=120; RC=15000
IF QC GT 0, SKIP 0
QC=360; RC=20000

THE ABOVE SEQUENCE FIRST SCALES SAM FROM HALF SCALE (BECAUSE RC IS ZERO INITIALLY UNLESS OTHERWISE SET) TO MUCH SMALLER IN TWO SECONDS, AND THEN UP TO ABOUT 2/3 SCALE IN SIX SECONDS, IN A SINUSOIDAL FASHION. NOTE THE WAIT FOR THE QC VARIABLE TO HIT ZERO BEFORE PROCEEDING. YOU MAY VERY EASILY DOLOOP SEVERAL OF THESE SEQUENCES WITH THE GENERAL TIMING BEING UNAFFECTED BY THE AMOUNT OF DOLOOPING, SOMETHING WE NEVER COULD DO BEFORE. NOTE THAT THE SYSTEM IMMEDIATELY STARTS TO DECREMENT THE TIME VARIABLES SO YOU SHOULD SET THE GOAL VALUES ON THE SAME LINE WHENEVER POSSIBLE.

THAT'S ALL FOR THE TIME BEING.........

// ************************************************MACROS

MACROS

WRITING MACROS IN GRASS

1. WHAT IS A MACRO ANYWAY??

A MACRO IN GRASS IS SIMPLY A SET OF COMMANDS. BEFORE GRASS HAD MACROS, YOU HAD TO TYPE IN ALL COMMANDS ONE BY ONE. THIS WAS OBVIOUSLY QUITE TEOIDIOUS WHEN LONG SEQUENCES HAD TO BE REPEATED SO WE SOON ALLOWED PEOPLE TO SAVE SETS OF COMMANDS ON THE DISK WITH NAMES SO THEY COULD BE RECALLED AT ANY TIME. MACROS, AT FIRST, WERE VERY MUCH LIKE PLAYER PIANO ROLLS, BEING ABLE TO STORE ONLY LONG SEQUENTIALLY EXECUTED LISTS OF COMMANDS WHICH WERE PLAYED FROM ONE END TO THE OTHER, WITH NO BRANCHING OR REPEATING. THIS USE OF MACROS IS STILL VERY COMMON AND VALID, SO HERE'S HOW TO DO IT.
1. INTERACTIVELY, BY TYPING A NAME, A COLON, AN OPEN BRACKET, THE COMMANDS WANTED, AND THEN A CLOSE BRACKET LIKE THIS:
   SAM:<GETDSK GLOBE
   ROTATE GLOBE, DO
   SCALE GLOBE, D1
   MOVE GLOBE, D2>
   THIS ACTION CAUSES A SMALL AMOUNT OF MEMORY (CALLED "CORE" BECAUSE THE MEMORY IS USED IN THE MEMORY) TO BE ALLOCATED WITH THE NAME SAM. "DO SAM" WILL CAUSE GRASS TO EXECUTE THE COMMANDS IN SAM ONE-BY-ONE UNTIL IT HITS THE END BRACKET. THEN GRASS RETURNS CONTROL TO WHATSOEVER ISSUED THE DO, WHETHER IT WAS YOU ON THE TELETYPE OR ANOTHER MACRO. NORMALLY, WHEN YOU ARE TYPING ON THE TELETYPE, YOU ARE IN WHAT WE CALL "STAR-LEVEL" (BECAUSE THE SYSTEM PRINTS **"S TO INDICATE A WAIT INSTEAD TO ACCEPT COMMANDS). NORMALLY, MACROS ARE STORED ON THE DISK FOR FUTURE USE BY THE FOLLOWING COMMAND:
   PUTDSK SAM
   SAM IS STORED ON THE DISK AS SAM.MAC THE .MAC IS AN EXTENSION AND IT TELLS YOU AND GRASS WHAT TYPE OF THING IS STORED ON THE DISK. A .DEC REFERS TO A GRASS PICTURE FILE (FROM DECIMAL) AND THE REST OF THE GRASS EXTENSIONS ARE GIVEN IN HELP HELP (JUST TYPE HELP HELP). NOW, TO RETRIEVE SAM AT A LATER DATE, SIMPLY TYPE DO SAM. IF SAM IS ALREADY IN MEMORY, IT WILL START TO EXECUTE. OTHERWISE, IT WILL BE Fetched FROM THE DISK UNDER THE NAME SAM.MAC. DISK WILL TELL YOU WHAT IS ON YOUR DISK AREA, YOU MAY ALSO TYPE SAM.MAC OR PRINT SAM.MAC OR GETDSK SAM.MAC IF YOU WISH. SEE THE RESPECTIVE HELPS (HELP TYPE, ETC.) FOR MORE INFORMATION.
   YOU CAN DELETE A MACRO BY TYPING DELETE SAM (RESUMING, OF COURSE, AS WE HAVE HERE, THAT SAM IS THE NAME OF YOUR MACRO). YOU CAN GET IT ERASED FROM THE DISK BY TYPING DELETE/D SAM.MAC.
   OFTEN, HOWEVER, ONE WANTS TO CHANGE A MACRO RATHER THAN TYPE IT ALL IN AGAIN. THE NEXT WAY OF ENTERING MACROS ALSO ALLOWS CHANGING.

2. USING THE EDITORS, THERE ARE TWO EDITORS IN GRASS--CALL EDIT AND EDIT MNAME--WHERE MNAME IS THE NAME OF ANY MACRO. CALL EDIT IS WORDY, ASKS YOU QUESTIONS AND HAS LOTS OF FEEDBACK. IT IS THE BEST EDITOR TO START WITH. YOU GET IT BY TYPING
   RESTART CALL EDIT
   IT WILL ASK YOU FOR THE NAME AND EXTENSION. IT ALSO ASKS FOR SOME INFORMATION ON WHAT THE MACRO IS TO BE USED FOR. AN AID TO YOUR OWN DOCUMENTATION. THIS INFORMATION IS STORED ON THE DISK UNDER MACROS.DOC WHICH YOU CAN TYPE OR PRINT ANYTIME.
   THE IDEA BEHIND EDITING MACROS IS VERY SIMPLE. MACROS ARE COMPOSED OF CHARACTER STRINGS AND THEY LOOK TO THE COMPUTER LIKE A BUNCH OF CHILDREN'S BLOCKS WITH LETTERS ON THEM. A COMMAND IN GRASS IS A SEQUENCE OF LETTERS (CHARACTERS) IN A ROW. ROTATE GLOBE, DO IS ACTUALLY STORED AS 19 BLOCKS IN MEMORY (ACTUALLY CALLED 'BYTES'). IF YOU COUNTED THE LETTERS AND SPACES, YOU GOT 17. THE TWO EXTRA ARE THE CARriage RETURN <CR> AND LINEFEED <LF> WHICH TELL GRASS THAT THE COMMAND LINE IS DONE. (BY CONTRAST, COMPUTER CARDS ARE ALL THE CHARACTERS, EVEN THOUGH MOST OftEN MOST OF THE SPACE IS BLANK, A WASTE OF SPACE AND TREES.) GRASS COMMANDS, THEN, ARE TERMINATED BY A CRLF SEQUENCE, WHICH, CONVENIENTLY, ALSO TELLS THE TELETYPE AND LINE PRINTER WHAT MECHANICAL ACTION TO PERFORM WHEN THE END OF THE LINE IS HAPPENING.
   SAY YOU HAVE A TEN-LINE MACRO CALLED SAM. THE BASIC THINGS YOU WANT TO DO IS ADD LINES (CALLED INSERTING) AND TAKE LINES OUT (DELETING). WITH CARD DECKS, THIS IS A PHYSICAL PROCESS, BUT IN GRASS, IT IS DONE ELECTRONICALLY AND ELECTROMAGNETICALLY. YOU HAVE TO GIVE THE EDITOR INSTRUCTIONS THAT LOGICALLY CORRESPOND TO ADDING AND PULLING CARDS FROM A DECK. GRASS EDITORS USE LINE NUMBERS SO YOU CAN EASILY TELL THE COMPUTER WHICH LINES TO DELETE OR AFTER WHICH LINE TO INSERT NEW LINES. NORMALLY, YOU ALSO WANT TO BE ABLE TO SEE THE CHANGES AS YOU MAKE THEM SO THERE'S A TYPE COMMAND AND A PRINT COMMAND, THE LATTER OF WHICH USES THE LINE PRINTER TO GET YOU HARDCOPY TO TAKE HOME AND SAVE.
   MOST PROGRAMMERS ARE TOO LAZY TO TYPE IN A WHOLE NEW LINE WHEN SOMETHING SIMPLE HAS TO BE CHANGED. SO WE ALLOW CHANGES TO BE MADE WITHIN A LINE WITH THE CHANGE COMMAND IN THE EDITOR. ONCE YOU CALL EDIT, AND ENTER THE MACRO NAMES, YOU CAN TYPE HELP TO GET SOME DOCUMENTATION ON THE COMMANDS AVAILABLE. YOU MIGHT AS WELL HAVE SOMEONE FRIENDLY HELP YOU EDIT THE FIRST TIME OR TWO.
   WHEN YOU ARE FINISHED EDITING, USE THE EXIT COMMAND TO GET THE MACRO CHANGED ON THE DISK. THE OTHER EDITOR, EDIT MNAME, IS A QUICKY, IN-CORE EDITOR. HELP EDIT DESCRIBES
3. Macros may also be constructed with string manipulation commands. This is pretty esoteric stuff which we mainly use to write programming super-languages in Grass or get around things the syntax of Grass cannot otherwise handle. See help strings when you're ready for this.

*************Note that macros, when stored on the disk, or entered with the editors, do not have the "NAME:<" at the beginning or the ">") at the end. Only the text is used, actually, and the brackets are to tell Grass that it's a macro only during the creation mode described in 1. above.

B. Macros as computer programs

We normally program computers using loops. Loops are sets of instructions repeatedly executed until some condition is satisfied. Grass does a lot of internal housekeeping to help the user get around writing trivial loops that graphics programming in Fortran, say, requires. The do loop method of running macros and the time-based variables as well as simple commands like rotate, scale and move all help set up things that normally require complicated loops in common graphics languages. Nevertheless, loops are very useful.

The command that explicitly causes looping is the skip command. It is exactly like the goto jail in monopoly or the chutes and ladders in chutes & ladders. Loops should normally have an ending condition, otherwise, an infinite loop results and you must exit with a control-c (hold the ctrl key down and hit the c at the same time, like a shift c--control-c is abbreviated [c]). [c] always gets you back to star-level. When a macro is executing, Grass will not accept commands from the teletype, with exceptions which you will discover on your own.

An example of a loop:

MOVE GLOBE. A, B, C
A=-10000
B=-5000
C=-2000
%LOOP A=A+50
B=B+25
C=C+10
IF A LT 15000, SKIP %LOOP

This loop will move the globe from the lower middle left of the screen to the upper middle right in a few seconds. When the value of A is 15000, the macro stops skipping backwards. You should be able to figure out the end values of B and C by now.

The if statement works as follows:


Note that any commands may be repetitively used in loops except things like restart, exit and goto which cause the control to be transferred out of the macro. Rotate, move and scale usually aren't used inside loops, just the variables need be changed inside the loop.

Macros can use other macros simply by saying do mname within a macro, the mname is executed and after it is finished, control returns to the statement after the one that had the do mname in it. If this is not clear at this point, ask someone to explain the concept of subroutines.

C. Getting macros to talk to you and listen to you

A. Talking or more properly, prompting:

The prompt command tells Grass that you want the value or contents of something typed on the teletype. Prompt do will tell the current value of do. Prompt 700/19 will give the value 36.84211. Similarly, prompt sqrt(FA*OA) will give a result depending on FA. Prompt also can type messages (hence its name) to the user, like:

Prompt "WHAT IS YOUR NAME"
OR
Prompt "THE VALUE OF A IS \ a
OR
Prompt "THIS MACRO HAS EXECUTED \ x \ "TIMES"
Any mix of variables and stuff in quotes (called string literals or just literals) may be prompted. Prompt is most often used in conjunction with the listen-to-you command, the Input command.

B. Input from the teletype

Input is one way of putting values in variables. The other way is direct assignment like:

A=10
K=K+500

Usually, in a macro, you want the user to type the value of some-
WOULD BE CLUMSY. THE INPUT COMMAND ALWAYS PRINTS OUT A QUESTION
MARK AND WAITS FOR SOMETHING TO BE TYPED AND CONTINUES TO WAIT
UNTIL IT SEES A CRLF(CARRIAGE RETURN). SO
BEFP:<INPUT A
PROMPT A**3.14159
SKIP -2>

WILL TYPE OUT THE AREA OF A CIRCLE WHOSE RADIUS YOU HAVE TO TYPE
IN UNTIL YOU HIT A C. TO STOP IT. THIS IS SIMPLER THAN TYPING
PROMPT A**3.14159 EACH TIME. LESS TRIVIAL LOOPS SHOULD OBVIOUSLY
BE POSSIBLE.

NORMALLY, THE MACRO SHOULD ASK FOR WHAT IT NEEDS IN THE IN-
PUT. FOR EXAMPLE:
BEFP:< PROMPT "WHAT IS THE RADIUS"|
INPUT A
PROMPT "THE AREA IS ", A**3.14159
SKIP -4>

THE "|" MAKES THE "?" FROM THE INPUT COMMAND APPEAR ON THE SAME
LINE AS THE PROMPT. YOU SHOULD NOW TRY SOME OF THESE SIMPLE EX-
AMPLES, BECAUSE IT'S GOING TO GET REALLY COMPLICATED HERE ON IN.

C. PROMPTING AND INPUTTING STRING VARIABLES.

OFTEN A MACRO IS WRITTEN TO BE GENERAL PURPOSE AND WORK ON
ANY PICTURE NAME. FOR INSTANCE, THE EARLIER MACRO FOR PLAYING
WITH THE GLOBE COULD BE REWRITTEN TO TAKE ANY PICTURE AS FOLLOWS:
NEWSAM: <PROM "WHICH PIX"

INPUT $A
GETDSK $A
ROTATE $A,Y,D0
SCALE $A,D1
MOVE $A,D2
PROMPT $A," IS MOVING ON D2,D3,D4; SCALING ON D1"

PROMPT "AND ROTATING AROUND THE Y-AXIS ON D0"

THIS IS AN EXAMPLE OF A MACRO THAT SAVES EFFORT. THERE ARE
SEVERAL GOOD REASONS FOR CREATING MACROS LIKE THIS: FIRST, YOU
CAN DO THINGS FASTER IF THEY CAN BE REPEATED AT COMPUTER RATHER
THAN HUMAN TYPING SPEEDS. SECOND, OTHER PEOPLE CAN WRITE MACROS
FOR YOUR USE AND VICE VERSA. THIRD, YOU CAN CALL THIS MACRO FROM
OTHER MACROS YOU WRITE AND BUILD UP A LIBRARY OF USEFUL SE-
QUENCES. EVENTUALLY, YOU WILL FIND THAT YOU NEVER DO ANYTHING
TWICE AND THAT'S GOOD.

BY THE WAY, IT'S BETTER TO CALL YOUR MACROS SOMETHING OTHER
THAN SAM AND NEWSAM. MACRO NAMES SHOULD BE SUGGESTIVE OF WHAT
THEY DO. NOTE THAT MACRO NAMES CAN ONLY BE SIX CHARACTERS LONG.

D. SOME POINTS:

1. MACROS ARE ENDED BY FALLING OFF THE END. NO EXPLICIT END
STATEMENT IS REQUIRED AS IN MOST PROGRAMMING LANGUAGES. SKIPPING
OFF THE END OF THE MACRO CAUSES IT TO RETURN TO THE MACRO IT WAS
CALLED FROM. THE RETURN COMMAND ACTUALLY FADES A SKIP 9999.
SKIPPING BACKWARDS MORE LINES THAN THERE ARE IS A CONVENIENT WAY
TO GET BACK TO THE BEGINNING OF A MACRO.

2. COMMANDS AND ALL NAMES IN GRASS (EXCEPT %LABELS) CAN BE ABBRE-
VIATED TO AS FEW LETTERS AS UNIQUELY IDENTIFY THEM. YOU WILL
SOON LEARN THAT R IS GOOD ENOUGH FOR ROTATE, AND SO ON, ALL BY
EXPERIMENTATION. YOU CAN CREATE THE NAMES SAM AND SAM, BUT NOT
IN THE ORDER SAM. SAM, BECAUSE THE SYSTEM ASSUMES SAM IS AN AB-
BREVIATION FOR SAM ONCE SAM HAS BEEN CREATED. YOU CANNOT HAVE
TWO THINGS NAMED THE SAME THING, NOR CAN YOU HAVE A MACRO AND A
PICTURE IN CORE WITH THE SAME NAME. THE SYSTEM WILL YELL AT YOU
FOR TRYING. THESE RESTRICTIONS DO NOT APPLY TO THE DISK NAMES.

3. MULTIPLE COMMANDS MAY BE PUT ON A SINGLE LINE BY SEPARATING
 THEM WITH SFM-COLONs. THIS IS VERY USEFUL FOR IF STATEMENTS.
 THERE ARE SEVERAL EXCEPTIONS: SEMI-COLONs MAY NOT BE USED AFTER
 DO'S OR CALL'S AND AFTER THE TEXT COMMAND. AN EXAMPLE:

PROMPT "HOW OLD ARE YOU?";INPUT B

4. STUDY SOME OF THE SIMPLE SYSTEM MACROS LIKE MERGE, JOIN,
 BIGGER, BIGGEST, AND SO ON. YOU CAN GET COPIES BY TYPING PRINT
 MERGE.MAC 31.3, ETC.

E. SOME FINE POINTS...

COMMENTS IN YOUR MACRO ARE PUT IN BY YOU TO TELL YOURSELF AT
A LATER DATE WHAT THE LOGIC BEHIND THE MACRO WAS. USE COM-
MENTS IF YOU DO NOT NOW, YOU WILL EVENTUALLY AFTER LEARNING
THE LESSON THROUGH CONSIDERABLE DUPLICATION OF EFFORT. A COMMENT
IS SIMPLY ANY LINE BEGINNING WITH A "**".

SEVERAL THINGS TO NOTE ABOUT COMMENTS:

1. LINES BEGINNING WITH A "#" ARE AUTOMATICALLY STRIPPED BY THE
 SYSTEM WHEN THE MACRO IS GOTTEN FROM THE DISK. THIS ACTION IS TO
 SAVE SPACE SINCE COMMENTS SHOULD BE AT LEAST AS MUCH TEXT AS COM-
 MANDS IN YOUR PROGRAM. CALL, EDIT, AND TYPING AND PRINT DO NOT STRIP
 THE
1. "*'s" means a macro is suspended temporarily, putting you into mode in which you can use any command. Typically, you prompt values of macros, or use the list or xlist command, or fix something gone wrong or otherwise missing, to get back into the macro, type resume.

2. '*'s" lines are not counted by the skip command (see help skip), but this is noteworthy only if you use numbers instead of %labels in your skips.

USE OF |s:
|s TEMPORARILY SUSPENDS EXECUTION OF A MACRO. THE SYSTEM PUTS YOU IN "*"-MODE IN WHICH YOU CAN USE ANY COMMAND. TYPICALLY, YOU PROMPT VALUES OF MACROS, OR USE THE LIST OR XLIST COMMAND, OR FIX SOMETHING GONE WRONG OR OTHERWISE MISSING. TO GET BACK INTO THE MACRO, TYPE RESUME.

THE COMPILER
THE COMPILER SPEEDS UP MACROS CONSIDERABLY. SEE HELP COMPILP AND HELP EXECUTE.

LOCAL VARIABLES
THE VARIABLES LA TO LZ ARE FIXED POINT VARIABLES WHICH ARE KNOWN ONLY TO THE MACRO THEY EXIST IN. THEY SHOULD BE USED WHENEVER POSSIBLE. THE SET EA TO EZ ARE FLOATING LOCALS. ALL OTHER VARIABLES ARE GLOBAL, WHICH MEANS ALL MACROS KNOW ABOUT THEM. THIS MAKES GLOBAL VARIABLES USEFUL FOR PASSING INFORMATION, BUT OCCASIONALLY CONFUSES VALUES. YOU WILL PROBABLY DISCOVER HOW AS YOU START TO PROGRAM IN GRASS. AT ANY RATE, YOU CAN REFER TO THE LOCAL VARIABLES OF A MACRO CALLED SAM FROM ANOTHER MACRO BY TYPING LA_SAM.

CLEVERLY PASSING VARIABLES
VARIABLES (NUMERIC AND STRING) MAY BE PASSED AS ARGUMENTS (STUFF WITH COMMAS AROUND) TO THE DO OR CALL OR EXECUTE COMMANDS. FOR EXAMPLE, YOU COULD USE THE MACRO NEWSAM FROM BEFORE BY TYPING:

```
DO NEWSAM,GLOBE
```

THE STUFF FOLLOWING THE MACRO NAME IS AUTOMATICALLY FED TO THE INPUT COMMANDS UNTIL IT RUNS OUT. IF THE MACRO HAS MORE INPUTS LEFT, THE PROMPTS, WHICH HAVE BEEN PUT TO SLEEP, SUDDENLY WAKE UP AND START ASKING QUESTIONS AGAIN. IF YOU WANT TO SUPPRESS THE LAST PROMPTS IN THE MACRO NEWSAM, TOSS IN AN EXTRA COMMA. YOU CAN ALSO USE THIS TECHNIQUE WITH SYSTEM MACROS:

```
CALL JOIN,THING,1000
```

OR
```
CALL EDIT,SAM,MAC, ••
```

YOU CAN ALSO PASS VARIABLES IN GLOBAL VARIABLES OR STRINGS, BUT THE MACRO HAS TO EXPECT THEM THAT WAY.

ERROR MESSAGES
GRASS'S ERROR REPORTING FACILITIES ARE PRETTY GOOD. WHEN YOU GET AN ERROR, THE SYSTEM STOPS, TYPES QUESTION MARKS, AN ERROR NUMBER AND TRIES TO LOCATE THE THING THAT CAUSED THE ERROR BY POINTING AT IT WITH A "|

IT THEN STICKS YOU IN "*"-MODE, JUST LIKE |s. YOU CAN FIX THE PROBLEM, |C TO EXIT, OR, IF YOU DON'T KNOW THIS ERROR NUMBER BY HEART YET (THERE'S ABOUT 200 YOU MIGHT RUN INTO), TYPE A "?" FOLLOWED BY A <CP> AND THE ERROR WILL BE EXPLAINED AS BEST WE CAN. MOST ERRORS ARE EITHER SPELLING ERRORS, RUNNING OUT OF CORE SPACE, OR NOT HAVING SOMETHING IN CORE THAT YOU REFER TO.
CREATING PICTURE LISTS IN GRASS

THERE ARE TWO BASIC WAYS TO CREATE PICTURES IN GRASS:

1. WITH THE EDITOR CALL EDIT
   USING CALL EDIT, YOU CAN INPUT A PICTURE BY TYPING *.DEC FOR THE
   EXTENSION. THE THING.DEC YOU CREATE IS STORED ON THE DISK
   AND MAY BE DISPLAYED WITH A GETDSK THING. THE POINTS THAT MAKE
   UP THE ENDPOINTS OF THE VECTORS YOU WISH TO DRAW MUST BE ENTERED
   AS FOLLOWS:
   
   X, Y, Z
   WHERE X, Y, AND Z RANGE FROM +2000 TO -2000. ALL THREE MUST BE
   SPECIFIED, ONE TO A LINE. THUS, A SQUARE IN THE UPPER RIGHT QUAD-
   DRANT OF THE SCREEN IS REPRESENTED BY:
   
   0, 0, 0
   1000, 0, 0
   1000, 1000, 0
   0, 1000, 0
   0, 0
   
   NOTE THAT FIVE POINTS ARE NEEDED TO DRAW FOUR LINES. THE SYSTEM
   ASSUMES LINES ARE DRAWN FROM THE PRESENT POINT TO THE NEXT POINT
   SPECIFIED. NOTE ALSO THAT THE ABOVE FORMAT ALLOWS SPACING AS YOU
   WISH SO FORTRAN 14 FORMAT ON CARDS WITH COMMAS BETWEEN IS ACCEPT-
   ABLE TO GRASS.
   
   TO GO TO THE NEXT POINT WITHOUT DRAWING A VECTOR (CALLED A
   MOVE-NO-DRAW), PUT A "J" ON THE LINES BETWEEN THE POINTS NOT TO
   BE CONNECTED. TO DRAW A "J", THE FOLLOWING IS ACCEPTABLE:
   
   0, 0, 0
   0, 1000, 0
   -500, 1000, 0
   500, 1000, 0
   
   AND SO ON. THE FIRST POINT IS ALWAYS ASSUMED TO BE A MOVE-NO-
   DRAW.
   
   AFTER TYPING THE POINTS YOU WANT, EXIT FROM THE EDITOR.

2. WITH PUTPOINT
   THE PUTPOINT COMMAND IS USED TO CONSTRUCT PICTURES IN MACROS. IT HAS A SIMILAR FORMAT TO THE DISK FORMAT ABOVE, BUT EN-
   CODES THE MOVE-NO-DRAWS DIFFERENTLY. HELP PUTPOINT SHOULD BE CON-
   SULTED FOR EXACT DETAILS. BRIEFLY, HOWEVER, THE PROCEDURE IS AS
   FOLLOWS:
   
   A. FIRST, YOU MUST OPEN THE PICTURE NAME. OPEN TELLS GRASS WHAT
      NAME THE ENSUING PUTPOINT'S, DELPOINT'S AND CLOSE WILL REFER TO.
      YOU MUST USE AN OPEN BEFORE PUTPOINT'ING.
      
      OPEN BOX
   
   B. THEN YOU TYPE IN, OR HAVE IN A MACRO:
      
      PUTP 0, 0, 0, 0
      PUTP 1000, 0, 0, 0
      PUTP 1000, 1000, 0, 0
      PUTP 0, 1000, 0
      PUTP 0, 0, 0
      
      THIS WILL DRAW THE BOX. NOW YOU MUST TELL THE SYSTEM TO CLOSE
      THE PICTURE OR IT CANNOT BE PUTSKID PROPERLY, AND YOU WASTE MOST
      OF THE PICTURE NUMBERS.
The fourth argument to PutPoint is commonly called \( K \) and specifies a draw if \( K = 0 \), and a move-no-draw if \( K = 1 \). So the "T" from before looks like:

\[
\text{OPEN TEE}
\]
\[
\text{PUTP 0,0,0,0}
\]
\[
\text{PUTP 0,1000,0,0}
\]
\[
\text{PUTP -500,1000,0,1}
\]
\[
\text{PUTP 500,1000,0,0}
\]
\[
\text{CLOSE}
\]

The numbers above may be replaced by variables or arithmetic expressions. For example, a macro to draw a series of 200 parallel vertical lines is:

\[
\text{LINES:<OPEN VERTS A=-10000 *SET BEGINNING POSITION OF A %LOOPY B=-10000 *SET B EACH TIME THROUGH LOOP PUTP A,B,0,1 MOVE-NO-DRAW TO FIRST POINT B=B PUTP A,B,0,0 DRAW THE LINE A=A+100 IF A LT 10000,SK %LOOPY CLOSE>}
\]

The arguments to PutPoint may be expressions but these are evaluated integer-wise, that is, with truncation to integer values after each part of the calculation (so \( .01 \times 1000 \) equals 0). You can use floating point calculations to get around this problem:

\[
\text{FA=.01*1000 FB=SIN(FK)*2000 PUTPOINT FA,FB,0,0}
\]

PutDSK VERTS will store this one on disk under VERTS.DEC. GetDSK VERTS will get it back (.DEC is the assumed extension for GetDSK).

The DelPoint command will erase the last PutPoint. A rubber band effect will be achieved by the following (the /16 is for scaling purposes):

\[
\text{PUTPOINT DO/16,D1/16,D2/16,0}
\]
\[
\text{IF FS1=0,DELPONT;SK -1}
\]
\[
\text{FSOFF 1;SK -2}
\]

A couple of things here—first, note that semicolons are used to put multiple commands on a line. Second, note the use of the function switches. To do this macro, you may want to try to add a function switch to cause move-no-draws, in which case you need a cursor moved on do,d1,d2 so you can see the move-no-draw point. You might want an exit function switch to do a close also. Note that to see the z-coordinate, you will have to rotate the picture too, and if you want the cursor to be attached, you will have to first group the picture and the cursor and rotate the group (not the pix). See help group for details. This type of thing gets mildly complicated, but is very instructive if you try it yourself.

C. Changing Endpoints with GetPoint and ZapPoint

Two commands, GetPoint and ZapPoint allow you to get at and change individual endpoints. They are complimentary and function like read and write. GetPoint gets values into variables and ZapPoint changes endpoints according to the values given in its variables. Help GetPoint and Help ZapPoint give all the straight info.

The special thing is that \( K \) (which can actually be any fixed variable) is set to 0 for draws, 1 for move-no-draws, and -1 for end-of-list. For example, a macro to make the first and last point of a picture the same (that is, close the gap between the first and last vectors) could be written as follows:

\[
\text{CLOSUP:<PROM "WHAT PIX TO BE DE-GAPPED"| INPUT $N N=1 GETP $N,1,A,B,C,D N=N+1 GETP $N,N,X,Y,Z,K IF K NE -1,SKIP -2 ZAPP $N,N,A,B,C,K PROM "DONE" >}
\]

An example to add 500 to each z if the x value is positive:

\[
\text{INCZ:<PROM "PIX NAME"| INPUT $A N=0 %MORE N=N+1 GETP $A,N,A,B,C,D IF A GE 0,C=C+500 ZAP $A,N,A,B,C,D FOR N NEW SKIP KOPES}
\]
GRASS uses simplified FORTRAN-style syntax for doing arithmetic. There are several important differences, though:

1. Variables in GRASS have fixed names (see HELP DEV). You cannot have a variable named "RATF", for example. This decision was made when designing GRASS and it eliminates the need to tell the system what type of variable (fixed, floating, string, array, etc.) your arbitrarily-named variable is. In GRASS, A-Z, VA-VZ, and WA-WZ are fixed point (integer) variables whose range (maximum and minimum values) is -32767 to 32767. Variable WZ normally has 32767 in it so if you need to know what full value of an integer variable is, prompt WZ.

FA-FZ are floating point variables with tremendous range. More computation is required to give this range, however, so one normally uses floating point variables (called reals in FORTRAN) only when doing fancy calculations. There are also variables known only within macros. These are called local variables and have the names LA-LZ (integer) and EA-EZ (floating). See HELP DEV for more info.

The rule is that the type of variable is indicated by its first letter. A list of variables is given in HELP DEV and HELP HELP.

2. There are two basic types of arithmetic statements: ones which integer destinations and ones which have floating destinations. For reasons of efficiency, assignment statements (ones with '=' signs) to integer variables (like A=...) are processed differently from assignments to floating variables. Floating assignments are more general; integer assignments are used for counting, etc.

Rule 1. When assigning to an integer variable, truncation to integer values is done after every step of the calculation. In floating assignments, the result is always kept to 8 significant figures. For example:

FD=.01
K=FD*100  evaluates to 0 since the .01 is TRUNCATED to 0 first.
FA=FD*100  evaluates to 10 since there is no TRUNCATION.

Rule 2. As explained in HELP DEV, only integer variables, or things that evaluate to integer variables may be directly attached to picture transformations (which is what GRASS as a language is most concerned with). You cannot rotate something on a floating variable without expressly converting it to an integer variable as in the following:

ROT GLOBE,Y,P
%LOOP FP=SIN(FA)*1000 P=FP
FA=FA+.01
SKIP %LOOP
THE P=FP CAUSES A FLOATING-TO-INTEGER CONVERSION.

Rule 3. Only floating assignments can use the built-in functions (SIN, COS, ATN, EXP, etc. explained below). This is also for reasons of efficiency.

Rule 4. All external control dev's (D0-D9, TX,TY,TZ, etc.) are integer and range from 32767 to -32768. Variable WZ is set to 32767 to help you use this silly number (32768 IS REALLY 2 TO THE 15TH POWER).

Rule 5. Because of digital wrap-around, 32767+1 evaluates to -32768. You will notice the effect of this if you try the following macro:

MOVE ANYPIX,A
A=A+100
SKIP -1
THE MEANING OF WRAP-AROUND WILL BE REAL CLEAR.

Rule 6. Don't put spaces around the '=' sign. Spaces anywhere else are OK though.

3. If you run out of variables, or need a way of specifying sequences of variables, use arrays (see HELP ARRAY).

4. Precedence of operators is as normal for algebra. You may use parentheses to change the order of precedence:

3+4*10 IS 43
(3+4)*10 IS 70

5. The prompt command can be used to type out the value of
6. Looping Arithmetic Statements Is Done Just Like All Other Loops In Grass. The Skip Command Is Used For Transfer Within a Macro and the Do Command Is Used To Call Other Macros As Subroutines. (See Help Skip, Help Do).

7. Compiling of Macros (See Help Compile) Speeds Up The Arithmetic Processing As Much As 200 Times. Compiling Is Recommended If Your Macro Has Lots Of Arithmetic and Is Running Too Slow For Your Tastes.

8. The Available Built-In Functions Are:
   1. $\sin(\text{EXPR})$ Where EXPR Is In Radians ($3.14159$ Radians Equals 180 Degrees).
      Ex: $\text{FA}=\sin(2.7*\text{FC})$
   2. $\cos(\text{EXPR})$
      Ex: $\text{FA}=\cos(\sin(2.7*\text{FC}))$
   3. $\arctan(\text{EXPR})$ Arc Tangent
      Ex: $\text{FK}=\arctan(\text{FE})$
   4. $\log(\text{EXPR})$ Log To The Base E
      Ex: $\text{FD}=\log(1.7E14)$
   5. $\text{ABS}(\text{EXPR})$ Absolute Value
      Ex: $\text{FF}=\text{ABS}(\text{FA})$
   6. $\text{RND}(\text{EXPR})$ Random Number
      Ex: $\text{FT}=\text{RND}(0)*1000$
      The Expr Is Included To Keep The Syntax Checker Happy But Is Not Used.
   7. $\text{SGN}(\text{EXPR})$ Sign
      Ex: $\text{FZ}=\text{SGN}(\text{RND}(0))$

Floating Point Numbers May Be Entered In E Format:
   $\text{FD}=1.2E7$
   $\text{FF}=-2E-10$

The Largest Positive Number Following The E Is 9808 And The Smallest Is -9808.

Use Trace, Prompt And List To Help Debug Your Arithmetic Programs.

That's All For Now.

The Explanation You Requested Does Not Exist (Check Spelling).

Array Array

Syntax:

array arrname,expr

or

array arrname,lower bound:upper bound

where lower and upper are expr's

The Grass Array Statement Is Like The Declare Statement In PL/1 And Like The Dimension Statement In Fortran. Its Purpose Is To Set Up The Array's Dimensions, That Is, How Big An Area In Core You Want To Set Aside For The Array.

Arrays Have A Special Set Of Names: AA Through AZ.

The Normal Use Of Array Assumes Your Array Dimensions Start At 1 And Go To The Expr. Consequently, Array Ab,10

Will Dimension An Array To Have 10 Integer (Like A-Z Type) Elements Referenced By AB(1), AB(2), ..., AB(10). In The Second Case, For Those Of You Who Must Have Unnatural Lower Bounds To Your Arrays, The Lower And Upper Bounds Must Be Expressly Stated. Note The Special Syntax--A ': ' Must Separate The Bounds.

Multi-Dimensional Arrays (Up To Four) May Be Created Too. You Simply Include The Dimensions For The Extra Columns In The Obvious Way:

Array A0,10,10,10

This Will Dimension A Floating Point Array 10X10X10.

Multi-Dimensional Arrays Use Lots Of Space, So Be Advised.

The Array Is Considered Fixed Point (Integer) If Its Name Is AA-AN. The Array Is Floating Point If The Array Name Is AC-AZ.

There Are Only Arithmetic Arrays; No String Arrays Exist In Grass.

Storing Arrays On The Disk Requires Assigning A Name With The Putdisk Command:

Putdisk Tiarr,ara,ak

Note The Syntax. The Array Name On Disk Is Tiarr,apa.

You Could Say Putdisk Ak,ara,ak But You Might Not Remember What Ak,ara Was For.

To Get It Back Into Core You Type:

Putdisk Tiarr,ara,ak
NOTE THAT IT CAN BE CALLED SOMETHING DIFFERENT THIS TIME, BUT YOU CANNOT GET FIXED AND FLOATING MIXED UP SO THE FOLLOWING WOULD BE UNWISE IN THIS CONTEXT:

GETDSK TIARR,A

SINCE GRASS WOULD ERRONEOUSLY THINK IT WAS A FLOATING ARRAY.

NOTE ALSO THAT YOU DO NOT USE THE ARRAY COMMAND PRIOR TO GETDSK!ING AN ARRAY--GETDSK DOES THE DIMENSIONING AUTOMATICALLY.

EXAMPLES:

THIS SEGMENT OF CODE SETS UP THE ARRAY, AND THE LOCOP FILLS IT IN WITH THE INPUT COMMAND.

```
ARRAY AB,300
A=0
XMORE A=A+1 ;*FIRST ELEMENT
PROM "ENTER ARRAY ELEMENT #",A|
INPUT AB(A)
IF A LE 300,SK %MORE
PROM "DONE"
```

EXAMPLE 2:

```
A=120
H=60
K=10
ARRAY AR(4:10),(A+H)/L
AR(4,0)=314159
AR(4,1)=2.789
AR(8,16)=.003
```

NOTE: OTHER OTHERWISE FILLED, ALL ARRAY ELEMENTS ARE SET TO 0.

```
// ********************************************************************
/* END OF A'S */
// ********************************************************************
// ****************************************************************** BACKUP
```

BACKUP SYNTAX:

```
BACKUP OLDNAME,NEWNAME
OR
BACKUP OLDNAME
```

SWITCHES:

```
/R
```

BACKUP CAUSES THE FILE SPECIFIED BY OLDNAME TO BE WRITTEN ONTO DECTAPE (UNIT 0). IF A SECOND ARGUMENT IS SUPPLIED THE NEW NAME ON TAPE WILL BE THAT NAME. IF NOT THE NAME ON TAPE WILL BE THE SAME AS THAT ON DISK. THE /R SWITCH REVERSES THIS TRANSFER CAUSING A FILE ON TAPE TO BE WRITTEN ONTO DISK. THE GENERAL FORM OF THE FilNAMES IS FilNAM.EXT #. # IF THE # IS NOT SPECIFIED THE DEFAULT IS THE AREA INTO WHICH THE USER IS CURRENTLY LOGGED. BE SURE THE DECTAPE IS ON REMOTE WITH THE WRITE ENABLE SWITCH ON AND THAT IT HAS BEEN INITIALIZED FOR YOUR AREA.

EXAMPLES:

```
BACKUP ROBIN.DEC
CAUSES THE FILE ROBIN.DEC TO BE WRITTEN ON TAPE WITH THE FILE NAME ROBIN.DEC
```

```
BACKUP ROBIN.DEC,JUNK.MAC
CAUSES THE FILE ROBIN.DEC TO BE WRITTEN ON TAPE WITH THE FILE NAME JUNK.MAC
```

```
BACKUP/R ROBIN.DEC
CAUSES THE FILE ROBIN.DEC ON TAPE ALREADY TO BE WRITTEN ON THE DISK. IT MUST NOT ALREADY EXIST ON THE DISK
```

```
// ******************************************************************BLANK
BLANK PIX
```

BLANK PIX ALLWS THE PIX TO BE UPDATED & GROUPED AND SO ON BUT CAUSES THE VECTORS TO BE UNDISPLAYED SO THEY DO NOT TAKE UP COMPUTER TIME TO DISPLAY. IT IS ALSO USEFUL FOR FLASHING A PIX OFF AND ON. IT IS USEFUL SOMETIMES, AS IN CERTAIN CASES OF PERSP, AND IN SYNCHRONIZING ROTATIONS TO NOT HAVE THE PIX PUT UP AND DOWN WHILE AT THE SAME TIME. A SETI PIX =32000 WOULD STILL POSSIBLY ADD TO FLICKER.

THERE IS ALSO A GETLIB/W OPTION NOW THAT ALLOWS YOU TO PUT UP A PIX OR PARTICULAR COMMAND HAVING ETC ON THE
FLASH YOU NORMALLY GET WHEN GETLIBING A PIX UNDER THESE CIRCUMSTANCES. GETL/W ESSENTIALLY DOES A BLANK AND THEN A BLANK/R AFTER 1/30 SECOND TO ALLOW ALL THE UPDATES TO TAKE EFFECT. IF YOU HAVE NEVER RUN INTO THIS PROBLEM, DON'T WORRY ABOUT IT, THOUGH.

SWITCHES: /R TURNS THE VECTORS BACK ON

EXAMPLES:
BLANK SAM
TICK 4
BLANK/R SAM
SK -3
THIS WILL FLASH SAM ON AND OFF 15 TIMES A SECOND (ONE SECOND EQUALS 60 TICKS).

// BLEND

SYNTAX: BLEND PIX1,PIX2,DEV,EXPR
BLEND TAKES TWO PIX AND DOES A LINEAR INTERPOLATION BETWEEN THEM. THE NUMBER OF STEPS IN THE INTERPOLATION IS GIVEN BY EXPR. THE STEP YOU ARE AT IS GIVEN BY THE DEV (WHICH FOLLOWS GRASS2 DEV CONVENTIONS). THE PIX SHOULD HAVE THE SAME NUMBER OF VECTORS FOR BEST RESULTS, ALTHOUGH THE WORST THAT SHOULD HAPPEN IS SOME RANDOM GARBAGE AT THE END OF THE SMALLER PIX OR LOSING SOME OF THE LARGER PIX. THE DEV VALUE NORMALLY RANGES FROM THE EXPR VALUE TO ZERO BUT MAY EXCEED THESE BOUNDS IF YOU WISH, IN WHICH CASE THE EFFECT IS WEIRD BUT OCCASIONALLY INTERESTING. BLEND, ONCE STARTED FOR TWO PIX, CONTINUES TO OPERATE UNTIL BLEND/U PIX1 IS TYPED. BLEND USES A LOT OF COMPUTER TIME, SO YOU DON'T WANT TO HAVE IT OPERATING WHEN YOU DON'T NEED IT. BLEND ALLOWS MANY PIX TO BE BLENDING INTO OTHER PIX AT THE SAME TIME, USING MORE COMPUTER TIME, NATURALLY. YOU MAY NOTICE THAT DIRCOR AND OTHER THINGS SLOW DOWN A LOT WHEN LARGE PIX ARE BLENDED. BLEND ALSO REQUIRES SCRATCH SPACE EQUIVALENT TO PIX1 WHILE OPERATING TOO, SO REMEMBER TO BLEND/U WHEN IT'S OVER. BLEND ALSO WORKS WELL WITH THE TIME-BASED VARIABLES.

EXAMPLES:
BLEND SAM,TCM,0.0,32767
THIS WILL BLEND SAM TO TOM ON DIALO
QA=400;BLEND SAM,TOM,QA,400
THIS WILL BLEND SAM TO TOM IN 400 STEPS TAKING JUST UNDER SEVEN SECONDS (SEVEN SECONDS = 420 TICKS). YOU MAY HAVE TO REVIEW THE TIME BASED VARIABLES TO UNDERSTAND THIS ONE.
BLEND SAM,TOM,100,200
THIS WILL BLEND SAM HALFWay TO TOM AND STAY THERE.
BLEND SAM,TOM,A,1000
A=A-10
IF A GT 0,SK -1
A=A+10
IF A LT 1000,SK -1
SK -4
THIS WILL BLEND TOM TO SAM AND BACK UNTIL STOPPED.

NOTE THAT A DEV VALUE = ZERO MEANS ALL THE WAY TO THE SECOND PIX, AND A DEV VALUE = THE EXPR MEANS PIX1 WILL LOOK LIKE PIX1.

SWITCHES: /U UNDO THE BLENDING
EX: BLEND/U SAM

// BUMP

SYNTAX: BUMP $VAR STRING MANIPULATION////
BUMP IS USED TO STEP THROUGH THE FIXED & FLOATING POINT VARIABLE NAMES (A-WZ,FA-FZ,AA-AZ,LA-LZ). THE IDEA IS TO ALLOW MACROS TO SETUP CODE TO MOVE, SCALE, ETC. A VARIABLE NUMBER OF PICTURES ON VARIOUS BINARIES. BUMP ONLY WORKS WITH $VAR'S WITH PROPER VARIABLE NAMES IN THEM, WHEN $A='K', BUMP $A WILL CHANGE THE 'K' TO 'L' AND SO ON. 'Z' GOES TO 'VA' AND 'VZ' GOES TO 'WA'.

EXAMPLE:
FROM "NUMBER OF COPIES"
INPUT N
M=0
$R='VA'
M=M+1
$A='COPY',M
COPY NAME,$A
SETI $A,$B
BUMP $B
IF M LF N,SK -5

THIS WILL
SETINT COPY1,VA
SETINT COPY2,VA
SETINT COPY3,VA
// ********************************************************************
/*
THERE IS NO DESCRIPTION AVAILABLE FOR THIS COMMAND AT THIS TIME, OR
COMMAND NAME HAS BEEN MISSPELLED. ------ SORRY
// ********************************************************************

CALL
SYNTAX: CALL MNAME

CALL IS LIKE 'DO' EXCEPT THAT IT REFERENCES THE COMMON AREA ONLY.
CALL, AS WELL AS OTHER COMMANDS, MAY BE USED AS AN ARGUMENT TO RESTART.
EXAMPLES: CALL MERGE, PIX1, PIX2
CALL EDIT
RESTART CALL EDIT
NOTE THAT CALL AUTOMATICALLY GETS THE COMPILED VERSION OF THE MACRO FIRST, IF IT EXISTS, TO FORCE USING THE MACRO VERSION (NON-COMPILED) TYPE
DO MNAME, MAC 31, 3

// ********************************************************************

CUTOFF
SYNTAX: CUTOFF PIX, DEV1, DEV2, DEV3, DEV4, DEV5, DEV6

CUTOFF IS A HARDWARE FUNCTION WHICH CUTS OFF THE PICTURE EITHER OUTSIDE OR INSIDE A RECTANGULAR BOUNDARY SPECIFIED BY THE SIX DEVS. THE DEVS REPRESENT THE X-HIGH, X-LOW, Y-HIGH, Y-LOW, Z-HIGH AND Z-LOW BOUNDARIES IN THAT ORDER. THE DEFAULT VALUES ARE:
CUTOFF PNAME, 32767, -32768, 32767, -32768, 32767, -32768
CUTOFF EXPECTS SEVEN VALUES. IF YOU SPECIFY JUST ONE DEV LIKE
CUTOFF PNAME, D1
YOU WILL GET D1, D3, D5 TO D2, D4, D6 AND D7. YOU SHOULD SET
D1, D3, D5 TO FULL POSITIVE AND D2, D4, D6 TO FULL NEGATIVE TO SEE THE PIX. THIS WILL SHOW PNAME AS IT APPEARS WITHIN THE BOUNDARIES SET BY THE DIALS. OF COURSE, ANY LEGAL DEVS ARE OK. (SEE HELP DEV).
THE /R SWITCH WILL SHOW THE PIX OUTSIDE THE BOUNDARIES.
NOTE: CUTOFF SEEMS TO SHIFT WHEN SETP IS USED.
ALSO: CUTOFF IS NOT SOFT1ABLE (SEE HELP SOFT). YOU SHOULD USE THE WINDOW COMMAND FOR REAL CLIPPING (SEE HELP WINDOW).
SWITCHES: /R OUTSIDE INSTEAD OF INSIDE
EXAMPLES: CUT GLOBAL, 0
THIS WILL LEAVE THE DEFAULTS FOR ALL BUT Z-LOW
SO IT WILL SHOW HALF THE GLOBE (ALL POSITIVE Z-POINTS)
CUT BOX 1, 0=-D0, D0=-D0, D0=-D0
CUT/R BOX 2, 0=-D0, D0=-D0, D0=-D0
THIS SEQUENCE WILL DO A CENTER WIPE FROM BOX 1 TO BOX 2 (FIGURE THAT ONE OUT).

// ********************************************************************

CLEAR
SYNTAX: CLEAR

CLEAR SIMPLY CLEARS THE VT05 SCREEN.

// ********************************************************************

CLOSE
SYNTAX: CLOSE

CLOSE ENDS THE PIX PREVIOUSLY SPECIFIED BY OPEN. CLOSE
ALLOWS THE PIX TO BE PUTFSDK'S, IT RELEASES ANY SPACE NOT NEEDED,
AND IT ALLOWS ANOTHER PIX TO BE OPEN'ED.
EXAMPLE:
OPENO SAM
PUTPOI 500, 500, 500, 0
PUTPOI -500, 500, 500, 0
PUTPOI 0, 0, 0, 0
PUTPOI 500, 500, 500, 0
CLOSE

// ********************************************************************

CLIP
SYNTAX: CLIP CLIPEE, CLIPER, CLIPPED
WHERE CLIPEE, CLIPER, AND CLIPPED ARE PIX
IT IN THE X AND Y DIRECTIONS SO THAT IT FITS INSIDE ANOTHER PICTURE, THE "CLIPPER", AND PUTS THE RESULT IN THE NEW PICTURE, THE "CLIPPED".

THE "CLIPPER" IS ASSUMED TO BE TWO-DIMENSIONAL, HENCE THE Z-COORDINATES ARE IGNORED. IT IS ALSO ASSUMED THAT THE "CLIPPER" HAS NO JUMPS IN IT, AND THEREFORE IF THERE IS A JUMP, A LINE WILL BE ASSUMED IN THE GAP. ALSO, AND THIS IS IMPORTANT, THE FIRST POINT OF THE "CLIPPER" MUST BE THE SAME AS THE LAST POINT. BASICALLY THE "CLIPPER" CAN BE ANY OBJECT WITH A CLEARLY DEFINABLE INSIDE AND OUTSIDE WHETHER IT BE CONCAVE OR CONVEX, SUBJECT ONLY TO THE ABOVE RESTRICTIONS.

SO FOR EXAMPLE, A FIGURE "8" DRAWN WITH A SINGLE BORDER IS ACCEPTABLE, BUT A FIGURE "8" DRAWN WITH A DOUBLE BORDER IS "BAD NEWS". THE OUTLINE OF A STAR IS ALSO ACCEPTABLE. HOWEVER, IF A LINE SEGMENT OF THE "CLIPPER" FACTUALLY COINCIDES WITH A LINE SEGMENT OF THE "CLIPPER", IT WILL BE EXCLUDED OR INCLUDED IN THE THE "CLIPPED" PICTURE DEPENDING ON THE JUDGMENT OF THE MOON AND STARS, ASSUMING THAT THE CORRECT ATMOSPHERIC CURRENTS ARE PREVAILING.

NOTE: THAT THE THE ORGINAL PICTURE IS LEFT UNMOLESTED, AND THAT THERE ARE NO RESTRICTIONS WHATSOEVER THE CLIPEE

*-NOTE: AT THIS POINT IN TIME THERE EXISTS AN IMPLEMENTATION RESTRICTION IN THAT THE CLIPPER MAY NOT CONTAIN MORE THAN 100 POINTS. THE REST OF THE BIG CLIPPER'S POINTS WILL BE IGNORED

SWITCHES: /R - FCP REVERSE CLIPPING
THE CLIPPED PICTURE WILL CONTAIN ONLY THE LINES AND PORTIONS OF LINES LEFT OUT IN ORDINARY CLIPPING. NOTE THAT IF THE CLIPPER DOES NOT FOLLOW THE ABOVE-MENTIONED RULES, CERTAIN ANOMALOUS LINES WILL BE LEFT CUT IN BOTH TYPES OF CLIPPING.

// *******************************************
// SYNTAX: COMPIL MNAME,BNAME
OR
COMPIL MNAME

COMPIL TRANSLATES MACROS INTO EXECUTABLE BINARY MACHINE INSTRUCTIONS, AND IS USED TO INCREASE THE EXECUTION SPEED OF A MACRO.
MNAME IS ANY MACRO AND IT IS CALLED INTO CORE AUTOMATICALLY IF NECESSARY. BNAME IS THE NAME OF THE .CPL FILE GENERATED AND IT CONTAINS THIS BINARY MACHINE CODE. IF BNAME IS NOT SPECIFIED, THE .CPL FILE IS NAMED THE SAME AS MNAME (MNAME THE MACRO IS AUTO-DELETED IN ANY CASE).
BNAME CAN BE SAVED BY USING: PUTDSK BNAME.CPL
(THE .CPL MAY BE OMITTED. PUTDSK IS SMARTER THESE DAYS.)
TO RUN .CPL FILES THE EXECUTE COMMAND CAN BE USED.

NOTE: DON'T TRY TO TYPE OR PRINT .CPL FILES, THEY ARE NOT IN THE SAME FORMAT AS OTHER FILES.
THE ONLY COMMANDS FULLY COMPILABLE ARE ARITHMETIC ONES, GETPOI, PUTPOI, ZAPPOL, SKIP, FLOATING POINT STUFF, AND IF.

NOTE: ALL COMMANDS CAN BE COMPILED. ALL COMMANDS NOT LISTED ABOVE ARE NOT CONVERTED INTO TRUE MACHINE INSTRUCTIONS, SO COMPILING DOES SPEED UP PROCESSING MUCH.

EXAMPLE 1:
GLOB;<A=A+1;SKIP 0>
COMPIL GLOB,TEST
PUTDSK TEST.CPL
EXECUTE TEST

EXAMPLE 2:
COMPIL GLOB
PUTDSK GLOB.CPL

// *******************************************
// SYNTAX: COPY PIX1,PIX2
COPY CAUSES PIX2 (THE NEW NAME) TO SHARE THE DATA OF PIX1 (THE OLD NAME). ANY COMMAND THAT DOES NOT MODIFY DATA LISTS (E.G. ROTATE, SCALE, MOVE, ETC.) MAY BE THEN INDEPENDENTLY BE USED ON EITHER PIX. COMMANDS THAT ALTER DATA LISTS (E.G. SMOOTH, SOFTROT) WILL MODIFY BOTH COPIES. COPY MAY BE USED TO REFLECT A PICTURE AROUND AN AXIS BY USING THE SINGLE DIMENSIONAL SCALE.
A SURE WAY TO CRASH THE SYSTEM IS TO DELETE THE FIRST PICTURE (PIX1) BEFORE DELETING THE COPY (PIX2)--SO DON'T DO IT.

WE NOW HAVE NEW SWITCHES FOR COPY, SOME OF WHICH ACTUALLY COPY THE VECTORS TOO. SO, IN THESE CASES (SEE SWITCHES BELOW), ALL THE WARNINGS ABOUT DELETING THE FIRST PIX AND SOFT AND SMOOTH, ETC. DO NOT APPLY.

SWITCHES: /T COPIES LIKE COPY BUT ALSO COPIES ALL THE
PUTL'S THE COPY
COPIES THE VECTORS TOO (BUT NOT THE
TRANSFORMATIONS
LIKE /V/T
LIKE /V/P
THERE ISN'T ANY /T/P (DO YOU WANT IT??)

EXAMPLE:
COPY GLOBE,WORLD
ROTATE GLOBE,X,D0,D1,D2
ROTATE WORLD,Z,D9
SCALE WORLD,D6

THE ABOVE SHOW INDEPENDENT MOVEMENT OF COPIES
COPY/V SAM,TOM
SC TOM,0
SOFT TOM
PUTD TOM

THE ABOVE PUTS A HALF-SIZED VERSION OF SAM ON THE
DISK UNDER THE NAME TOM

CREATIVE...

SYNTAX:
CRUNCH CPLNAME,PNAME,EXPR1,EXPR2,XVAR,YVAR,ZVAR

CRUNCH IS AN AUTOMATIC GETPOI/ZAPPOI COMBINATION. IT EXPECTS
A COMPILED MACRO AS ITS FIRST ARGUMENT WHICH IS SUPPOSED TO AFFECT
IN SOME WAY THE XVAR, YVAR AND ZVAR SPECIFIED BETWEEN THE
POINT RANGE EXPR1 TO AND INCLUDING EXPR2.

EXAMPLE:
FOO:<X=X*2
Y=Y*3
Z=X+Y>

THE ABOVE WILL DO THE SAME AS:
N=4
N=N+1
GETPOI CIRC9,N,X,Y,Z,K
ZAPPOI CIRC9,N,X*2,Y*3,X+Y,K
IF N LT 100,SKIP -3

EXCEPT THAT CRUNCH IS MUCH MUCH FASTER AND TAKES INTO ACCOUNT
CASES WHEN EXPR2 IS GREATER THAN THE NUMBER OF POINTS.
TO APPLY THE CPLNAME TO ALL THE POINTS, SPECIFY 9999 AS EXPR2.
THE CPLNAME CAN, OF COURSE, BE STORED ON THE DISK; CRUNCH GETS
IT AUTOMATICALLY.

CORE

SYNTAX:
CORE

CORE PRINTS OUT THE TOTAL NUMBER OF FREE WORDS OF CORE
AVAILABLE AND ALSO PRINTS THE SIZE OF THE FRAGMENTS. IF VAR1
IS SPECIFIED IT GETS THE TOTAL NUMBER OF FREE WORDS. IF VAR1 AND VAR2
ARE SPECIFIED VAR1 GETS THE TOTAL NUMBER OF FREE WORDS AND VAR2 GETS THE
SIZE OF LARGEST FRAGMENT.

DELETE

SYNTAX:
DELETE ANAME

DELETE ANAME FROM CORE, REMOVES THE NAME AND RECLAIMS THE
STORAGE THE ANAME TOOK.

SWITCHES:
NONE
/D DELETES ON YOUR DISK AREA (REQUIRES EXTENSION)
/IT DELETES ON YOUR TAPE AREA

SOURCES

SYNTAX:
CRUNCH CPLNAME,PNAME,EXPR1,EXPR2,XVAR,YVAR,ZVAR
EXAMPLES:
DELETE GLOBE
DELETE JSDRAW
DELETE/D WITCH.DEC
DELPOI
SYNTAX: DELPOI
OR DELPOI EXPR
DELPOI DELETES THE LAST PREVIOUSLY PUT POINT IN AN OPENED PICTURE. THE NAME OF THE PICTURE IS AUTOMATICALLY ASSUMED TO BE THE LAST PICTURED OPENED.
IF AN EXPR IS INDICATED, THAT NUMBER OF POINTS IS BACKED UP.
EXAMPLE:
<PUTPOI DO/16, DO/16, DO/16, 0
IF FS1=!, DELPOI
SKIP -2>
THIS CAUSES VECTORS TO BE DRAWN FROM THE POSITIONS OF DIALS 0, 1, AND 2 UNLESS FS1 IS BEING HELD DOWN, IN WHICH CASE, A RUBBER-BAND EFFECT IS ACHIEVED.
EXAMPLE: DEMP 5
THIS WILL DELETE THE LAST FIVE POINTS IN AN OPENED LIST.
DELPOI
DELPOI
SYNTAX: DIRCORE
DIRCORE GIVES THE USER A MAP OF WHAT'S GOING ON IN USER MEMORY. PICTURES, MACROS, CPL'S, ARRAYS AND SO ON ARE LISTED WITH NOTES ON HOW MUCH SPACE THEY TAKE UP. THE TRANSFORMATIONS THAT ARE CURRENTLY ATTACHED TO PICTURES ARE ALSO LISTED.
DIRSK
SYNTAX: DIRSK (OPTIONAL NAME) (OPTIONAL AREA)
DIRSK GIVES A LIST OF STUFF THAT'S STORED ON THE DISK. THE OPTIONAL NAME (SEE EXAMPLES) ALLOWS ONE TO LOOK AT PART OF THE DIRECTORY. IF THE ** FEATURE ON IS COMBINED WITH THE ** FEATURE, ONE CAN SELECT PATTERNS OF AREAS OR NAMES.
IF THE OPTIONAL AREA IS SPECIFIED, IT MUST BE A VALID LOGIN AREA ON THE DISK.
SWITCHES:
/L DIRECTORIO IS LISTED ON THE LINE PRINTER INSTEAD OF THE VT05.
EXAMPLES:
DI GIVES YOU A LOOK AT YOUR DISK DIRECTORY
DIRSK *.DEC 1,1 LISTS ALL NAMES WITH A .DEC EXTENSION IN AREA 1,1
DIRSK S** LISTS ALL NAMES BEGINNING WITH S
DIRSK P** M** LISTS ALL NAMES BEGINNING WITH P AND EXTENSIONS BEGINNING WITH M
DIRTAP
SYNTAX: DIRTAP (OPTIONAL NAME) (OPTIONAL AREA)
DIRTAP GIVES A DIRECTORY LISTING OF A DECTAPE ON THE VT05 SCREEN. THIS HAS THE ** FEATURE SAME AS THE DIRSK COMMAND.
DEFAULT AREA IS THE USERS OWN TAPE AREA. BE SURE WHEN USING THIS COMMAND THAT THE DECTAPE IS ON REMOTE AND SET FOR UNIT 0.
EXAMPLES:
DIRTAP
THIS GIVES A DIRECTORY LISTING OF EVERYTHING ON THE USERS TAPE AREA.
DIRTAP *.DEC 1,1
THIS GIVES A DIRECTORY LISTING OF ALL FILES WITH A .DEC EXTENSION IN TAPE AREA 1,1.
DO
SYNTAX: DO MNAME
OR DO MNAME.MAC XX, XX
OR DO MNAME,ARG1,ARG2,.......
DO IS USED FOR EXECUTING DISK- OR CORE-RESIDENT MACROS.
THE EXTENSION AND DISK AREA (XX XX) MUST BE INCLUDED IF YOU
DELETION SUCH.

ARE CALLING A MACRO FROM SEVERAL LEVELS 6 CALL-DOMPED MACRO COUNT AS OPERANDS TO THE RESTART COMMAND AND ISSUED WITHIN OTHER
MACROS IF DESIRED. ARGUMENTS MAY BE PASSED TO THE MACRO BY THE
USE OF ARG1, ARG2, .... (READ BY THE INPUT COMMANDS INSIDE THE MACRO).
IF ANOTHER MACRO IS CALLED THE OLD ARGUMENTS ARE LOST.

EXAMPLES:
DO FLUFF
DO FLUFF, MAC 30, 4
DO FLUFF, MAC 31, 3
DO FLUFF, 10, 22, SAM, DEC
10 IS PASSED TO FIRST INPUT
22 IS PASSED TO SECOND INPUT
THE STRING SAM, DEC IS PASSED TO
THE THIRD INPUT COMMAND. ANY
FURTHER INPUTS WILL GO TO THE
KEYBOARD.
NOTE THAT PROMPTS ARE SUPPRESSED
UNTIL THE ARGS RUN OUT.

DCLOOP

SPECIAL MACRO COMMANDS

DCLOOP
SYNTAX:
DOLOOP MNAME 1, MNAME 2, ....  ESOTERIC//
OR               DOLOOP MNAME
OR               DOLOOP <

DOLOOP IS AN ALTERNATIVE TO 'DO' AND IT RUNS THE MACRO
(OR .CPL) AS A BACKGROUND JOB. WHEN USING DCLOOP, ALL MACROS
SHOULD BE DOLOOPED SINCE *LEVEL & NORMAL MACRO LEVEL HAVE
PRIORITY. THINGS REQUIRING INPUT FROM THE TERMINAL IN A DCLOOPED
MACRO WILL STOP ALL MACROS, AS WILL SWAP MODULES WHILE THEY ARE
SWAPPING IN. DOLOOPED MACROS MAY HAVE SKIPS; BUT A SKIP AT THE END
BACK TO THE BEGINNING IS ASSUMED. TICK WILL WORK INDIVIDUALLY FOR
EACH MACRO. LOCAL VARIABLES SHOULD BE USED WHENEVER POSSIBLE.
AT LEAST BE VERY CAREFUL WHEN USING VARIABLES WITH DCLOOPING SC
YOU DON'T HAVE TWO MACROS CHANGING THE SAME VARIABLE UNINTENTIONALLY.
DOLOOPED MACROS ARE EASILY STOPPED BY IC.

SWITCHES:
DOLOOP/E MNAME, EXPR
DO THE MACRO EXPR NUMBER OF TIMES
DOLOOP/V MNAME, VAR
DO THE MACRO UNTIL # OF TIMES = VAR

EXAMPLES:
DOLOOP <D=D+05/32>
DOLOOP SAM, FIRED, COMPACT

NAMED DOLOOPED MACROS MAY BE INDIVIDUALLY CANCELLED BY UNLCOP.

EXECUTE

SYNTAX:
EXECUTE BNAME  ESOTERIC//
OR          EXECUTE BNAME, ARG1, ARG2, ........

EXECUTE IS TO .CPL (COMPILLED) MACROS WHAT "DO" IS TO
NORMAL MACROS. THE COMPILLED MACRO IS GOTTEN FROM THE DISK (YCUR AREA
UNLESS SPECIFIED IN, X, Y LOGIN CONVENTIONS) UNLESS IT'S ALREADY
IN CORE, AND THEN THE COMPILLED MACRO IS RUN. THERE IS NO
EQUIVALENT TO GOTO FOR COMPILLED MACROS, BY THE WAY.
IC AND IS CAN STOP EXECUTION OF COMPILLED MACROS.

EDIT

SYNTAX:
EDIT MNAME

EDIT IS AN IN-CORE QUICKY EDITOR WHICH WORKS ONLY ON MACROS.
IT IS FAST, SMALL, AND HAS THE ADVANTAGE THAT YOU DO NOT HAVE TO RESTART
GRASS EACH TIME YOU WANT TO USE IT.  HOWEVER, IT CAN BE ENTERED AND
RE-ENTERED VERY QUICKLY.
IT'S MAJOR DISADVANTAGE IS THAT IT DOES NOT DO EVERYTHING
THAT CALL EDIT DOES.  FOR INSTANCE, YOU TYPE IC TO GET OUT OF THIS
EDIT, AND THE MACRO IS UPDATED ONLY IN CORE, AND NOT ON THE DISK.
FOR THIS REASON, WE HAVE IMPLEMENTED A NEW PUTDSK SWITCH--PUTDSK/D--
WHICH PUTS THE FILE ON THE DISK, FIRST CREATING A BACKUP
(.BAK) IF THE FILE ALREADY EXISTS ON DISK.  THEREFORE, AS SOON AS
YOU HAVE A WORKING VERSION OF A MACRO, PUTDSK/D IT OR YOU WILL loose
IT UPON RESTART.  ONE MORE THING:  THIS EDIT WILL NOT CREATE MACROS.
IT WILL ONLY WORK ON MACROS THAT EXIST ON DISK OR IN CORE ALREADY.
EDIT WILL FETCH A MACRO FROM THE DISK IF IT IS NOT ALREADY
IN CORE, BY THE WAY.

EDIT COMMANDS
TO TYPE THE WHOLE MACRO
TO TYPE LINES 100 TO 100<CR>
TO TYPE LINES 50 TO 200
TO DELETE LINE 20
TO CHANGE AAA TO JHKL IN LINE 80
TO CHANGE ALL OF LINE 400 OR 80<CR>
TO INSERT AFTER LINE 30
TO EXIT FROM EDIT

TO TYPE LINE 100
50<CR>
20<CR>
80<CR>
80<CR>
80<CR>

400 THIS IS A NEW LINE<CR>
31 THIS IS A NEW LINE<CR>

SOME NOTES:
1. THIS EDIT PENUMBERS THE LINES AFTER EACH INSERT OR
DEDELETE. THIS MEANS THAT YOU SHOULD LIST THE LINES YOU WANT
TO PLAY WITH AFTER EACH INSERT OR DELETE BECAUSE THE LINE NUMBERS
WILL SHIFT. HOWEVER, YOU CAN MULTIPLY DELETE LINES
20 THRU 40 BY TYPING "20<CR>" THREE TIMES (TRY IT).
YOU CAN ALSO MULTIPLY INSERT AFTER LINE 20 BY TYPING:
21 THIS IS LINE AFTER 20
31 THIS IS THE NEXT LINE
41 THIS IS THE NEXT LINE
ETC. THIS MAY LOOK WEIRD BUT IT'S EASY TO GET USED TO.

2. LOCAL VARIABLE TABLES ARE ZEROED BY THIS EDITOR. OTHERWISE, IT
IS POSSIBLE TO EDIT MACROS OVER AND OVER AGAIN WITHOUT
RESTARTING, WITH NO SIDE EFFECTS.

3. IF YOU ARE TIGHT ON SPACE AFTER USING THIS EDIT, DEL/C EDIT TO
GET RID OF IT
// **-----------------------------------------------------------------------
EXIT

EXIT IS SIMPLY A PROGRAMMED |C. IF YOUR MACRO SEES AN
"EXIT" COMMAND, THE SYSTEM RETURNS TO *-LEVEL DOING WHATEVER A |C WOULD.
// **-----------------------------------------------------------------------
/*
WE DON'T HAVE THAT ONE AROUND...COMPLAIN LIKE HELL
// **-----------------------------------------------------------------------
*/
**-----------------------------------------------------------------------

FILMING SYNTAX:

FILMING
FILMING EXPR
FILMING/M EXPR

FILMING IS A COMMAND WHICH ALTERS THE SPEED OF THE UPDATES
OF ROTATIONS, MOVES, SCALES, TIME-BASED VARIABLES, DIAL
READINGS, VIPED CPL'S, ETC. FILM (WITHOUT ANY EXPR) STOPS
UPDATES AND WILL CONTINUE ONLY WHILE FS1 IS HELD DOWN.
"FILM EXPR" CAUSES THE UPDATES TO BE DONE WITH
TIME DELAYS EQUAL TO THE NUMBER OF TICKS SPECIFIED BY THE EXPR.
SO FILM 2 WILL UPDATE EVERY OTHER 1/60 SEC (30 TIMES A SECOND)
AND FILM 60 WILL UPDATE ONCE A SECOND. THIS COMMAND
FEATURE IS USEFUL TO CAUSE MOTIONS TO APPEAR TO STOP, OR, IS OCCASIONAL
USEFUL IF TOO MUCH BLENDING, ROTATIONS ON GROUPS, ETC.,
CAUSE THINGS LIKE DIRCOR OR PUTDSK TO OPERATE TOO SLOWLY.

FILM/M EXPR IS USED TO TRIGGER THE MOVIE CAMERA SITTING
ON TOP OF THE SCOPE. YOU SHOULD TALK TO TOM ABOUT USING THE
CAMERA SINCE IT IS A PRIMITIVE, SILVER-EATING THING
WHICH IS NOT REAL-TIME IN OPERATION. FILM/M DOES A
PRETTY GOOD JOB OF SLOWING DOWN PROPERLY WRITTEN MACROS
SO IMAGES RECORD WELL ON FILM.

EXAMPLES:
FILM
FILM/M 100
// **-----------------------------------------------------------------------
EXIT

FIX SYNTAX:

FIX PNAME

FIX FREEZES THE POSITION OF A PICTURE OR THE VALUE OF A PNAME'S
MODIFIER ACCORDING TO THE SWITCH OPTIONS. NO SWITCH FIXES EVERYTHING.
FIX ALSO REMOVES DEV ASSIGNMENTS IF ANY.

SWITCHES:

NONE
/P
/S
/M
/P
/I
/O
/C
/D

EXAMPLES:
/FIX'S OPTION
**FSOFF**

**SYNTAX:**

```
FSOFF EXPR1,EXPR2,EXPR3, ETC.
```

FSOFF turns off the function switches corresponding to the expressions. FSOFF alone turns off all the switches.

**EXAMPLE:**

```
FSOFF 1,3,5,7,9,11,13,15
```

This turns off the odd function switches.

```
FSOFF
```

This turns off all the switches.

---

**FSON**

**SYNTAX:**

```
FSON EXPR1,EXPR2,EXPR3, ETC.
```

FSON turns on (sets to one) function switches corresponding to EXPR1,EXPR2, ETC. FSON does not work with the panic button.

**EXAMPLE:**

```
FSON 0,2,4,6,8,10,12,14
```

This lights up the even switches and sets them to equal 1.

---

**GETDSK**

**SYNTAX:**

```
GETDSK DNAME.EXT XX,XX
OR
GETDSK DNAME.EXT XX,XX ,EXPR1,EXPR2,VAR
```

GETDSK gets the DNAME from the disk area indicated by "XX,XX" (default is your area). If no .EXT, it defaults to .DEC. DNAME becomes the name of the thing you are GETDSK'ing. If EXPR'S are given, they get lines starting at line EXPR1 up to and including line EXPR2 in the file. It returns a value of 1 in VAR if there is more and a value of 0 if end of file. GETDSK with EXPR'S is esoteric/////

**EXTENSIONS:**

- .DEC: Standard picture mode (default)
- .CPL: Complied format (for binary code. Don't type or print these files. Weird things happen.)
- .MAC: Macro format (for macros, not for pictures)

Note that .DEC type pictures come up on the screen.

**SWITCHES:**

- /P: (for .DEC only) picture is putlib'd first. With this switch, one can get a picture, rotate it, etc., and then GETL/W it so it will appear in the right place when first seen by whoever is in front of the VG.
- /M: ignores extension checking and gets the file as if it were a macro. It also allows remarks to be left in the macro. This is useful for handling .DEC files in ASCII.

**Note:** GETDSK normally strips off all lines beginning with the character '*' in .MAC files, so comments may be liberally provided in your macro without worrying that the macro will take up too much space.

**EXAMPLES:**

- GETDSK WITCH
- GETDSK WITCH.DE
- GETDSK WITCH.DE 30,10
- GETDSK DRAW.MAC
- G C AFUL.CPL
- G FORD.ARA,AH
  
(Puts FORD.ARA into ARRAY AH
No dimensioning necessary.
Note that the ARRAY command is only for setting up new arrays.)

- G/P BLOB
- R BLOB,Y,DO
- SC BLOB,15000
- MCV BLOB,TX
- GETL/W BLOB
  
Auto-putlib feature
Used intelligently.

---

**GETHIT**

**SYNTAX:**

```
GETHIT DNAME,XX,XX
```

---
GETHIT RETURNS THE STATUS OF A LIGHTPEN HIT WHERE N GETS THE NUMBER OF THE POINT IN PIX, X,Y,Z GET THE RESPECTIVE COORDINATES, AND K IS SET TO ZERO IF IT IS A DRAWN POINT, ONE IF IT IS THE INITIAL POINT OF A DRAWN VECTOR (I.E., FOLLOWING A JUMP), AND MINUS ONE IF IT IS THE LAST POINT IN THE PIX.

EXAMPLE:
GETHIT PANAM,F,U,C,K,Q
(AN EXAMPLE IN THE HUMOR STYLE OF MAINE EAST.)

GETLIB

SYNTAX:
GETLIB PNAME

GETLIB RETRIEVES THE PNAME FROM THE NON-DISPLAYED IN-CORE PICTURE LIST AND DISPLAYS IT. IF PNAME2 IS INDICATED, IT IS PUT IN THE TREE AFTER PNAME1 (SEE HELP TREE, HELP GETLIB). IF PNAME2 IS NOT SPECIFIED, PNAME BECOMES THE FIRST ELEMENT TREE.

NOTE THAT PNAME2 MUST BE DISPLAYED AND PNAME MUST HAVE BEEN PUTLIB'ED AT SOME TIME (BY PUTLIB OR GETDSK/P OR COPY/P). GETLIB AND PUTLIB ARE THE ONLY WAY TO MOVE THINGS AROUND IN THE TREE STRUCTURE.

EXAMPLES:
GETLIB SAM
GETLIB MARY,SAM

GETPOIN

SYNTAX:
GETPOIN PIX,N,X,Y,Z,K

GETPOIN GETS A POINT FROM AN LNAME. THE X,Y,Z COORDINATES OF THE NTH POINT ARE RETURNED IN VARIABLES INDICATED HERE BY X,Y,Z. N RANGES FROM THE 1 (THE FIRST POINT) TO THE LAST POINT IN THE LNAME (WHICH DEPENDS ON HOW BIG THE PICTURE IS). K IS A VARIABLE IN WHICH THE FOLLOWING IS INDICATED:

K=0 DRAWN VECTOR
K=1 NON-DRAWN VECTOR (JUMP) THAT IS, NO LINE DRAWN TO THIS POINT FROM THE LAST POINT.
K=-1 END OF LIST (LAST VECTOR)

SWITCHES:
/N GETS POINTS SEQUENTIALLY FROM A PICTURE. ONCE GETPOIN IS EXECUTED, THEN /N GETS NEXT POINT(S) & USES THE SAME VARIABLES. THIS HAS NO ARGUMENTS. IT ALSO SPEEDS UP PROCESSING GREATLY.

EXAMPLE:
GETPOIN WITCH,20,G,H,I,K

THIS WILL GET THE COORDINATES IN DECIMAL OF THE TWENTIETH VECTOR IN WITCH AND PLACE THEM IN VARIABLES G,H,I AND INDICATE WHETHER THIS LINE WAS DrawN, A JUMP OR THE END OF THE LIST, IN VARIABLE K.

THE APPLICATION FOR THE SWITCH IS:
<N=0
N=N+1
GETPOIN FL,N,X,Y,Z
SK -2>

THIS MACRO WILL GET ALL THE POINTS IN THE PIX CALLED FL, BUT IT CAN BE DONE FASTER WITH THIS MACRO:
<br><br><br>GETPOIN FL,N,X,Y,Z
GETPOIN/N
SK -1>

IN BOTH CASES, THE MACRO WILL GIVE AN ERROR MESSAGE AT THE END OF THE PIX LIST. THE SECOND WILL EXECUTE MUCH FASTER SINCE IT ALREADY KNOWS WHICH VARIABLES TO USE. GETPOIN/D DOES NOT WORK WELL WITH DOLOOPING, THOUGH.

A WAY TO GET TO THE END WITHOUT GETTING AN ERROR MESSAGE, BY THE WAY, IS:
<N=0
N=N+1
GETPIX N,X,Y,Z,K
IF K=1,SK-2
PRINT "PIX HAS "N, VECTORS"

GOTO

SYNTAX:
GOTO MNAME+EXPRESSION

GOTO IS USED TO TRANSFER CONTROL TO A MACRO WHEN YOU NEVER WANT TO RETURN TO THE MACRO THAT THE
DON'T DO IT UNLESS YOU REALLY HAVE TO.

GROUP

SYNTAX:

GROUP PNAME1, PNAME2, GNAME

GROUPS PNAME1, PNAME2 AND EVERYTHING BETWEEN THEM INTO A GROUP NAMED GNAME. PNAMEs MUST BE DISPLAYED. PUTLIB/GETLIB SEQUENCES MAY BE USED TO ALTER GROUP STRUCTURE BY ELIMINATING OR ADDING PICTURES IF NECESSARY. ANY COMMAND WHICH WILL WORK ON PNAMEs WILL WORK ON GROUPED PICTURES (EXCEPT DELETE). CHECK "TREE" BEFORE GROUPING TO MAKE SURE THE PNAMEs ARE ON THE SAME LEVEL & THE RIGHT STUFF IS GROUPED.

GROUP WORKS WITH BLANK'D PIX TOO.

THE COMMANDS WHICH WORK WITH GROUPS ARE:

GROUP CUTOFF FIX GETLIB MOVE PATHMOV PUTLIB RESET

WITH GETLIB AND PUTLIB, YOU CAN MOVE PIX & GROUPS AROUND IN THE TREE DATA STRUCTURE.

FOR EXAMPLE, IN:

GETD PLANE
GETD PROP
GROUP PROP, PLANE, SAM

THE TREE LOOKS LIKE:

<table>
<thead>
<tr>
<th>PROP</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLANE</td>
</tr>
</tbody>
</table>

SAM

NOW A PUTLIB PROP WILL UN-DISPLAY IT AND TAKE IT OUT OF THE GROUP SAM. A GETLIB PROP WILL RE-DISPLAY IT BUT IT WILL NOT BE IN SAM. TO GET IT BACK IN SAM, YOU HAVE TO GETLIB PROP, PLANE. FURTHERMORE,

GETD/P WHEELS
GETL/W WHEELS, PLANE

WILL GET WHEELS UNDER SAM WITHOUT ANY UNSIGHTLY FLASHING (DUE TO THE GETL/W WHICH WAITS FOR THE TRANSFORMATIONS TO TAKE EFFECT BEFORE DISPLAYING).

HELP

HELP SYNTAX: HELP COMMAND-NAME

HELP IS HERE TO ANSWER SYNTAX QUESTIONS.

CERTAIN ABBREVIATIONS HAVE BEEN USED WHICH WILL, HOPEFULLY, AID UNDERSTANDING:

ANAME IS ANY PICTURE OR MACRO NAME
PNAME IS ANY PICTURE NAME (GROUP OR SINGLE PICTURE)
PIX IS ANY SINGLE PICTURE
GNAME IS ANY GROUP NAME
MNAME IS ANY MACRO NAME
DNAME IS ANY DISK FILE NAME
ARRNAME IS ANY ARRAY NAME
CPLNAME IS ANY COMPILED MACRO IN BINARY FORM

<CR> IS THE SYMBOL FOR CARRIAGE RETURN, THE KEY YOU MUST PUSH TO END LINES YOU TYPE ON THIS TERMINAL.

NAMES IN CORE MAY BE ABBREVIATED TO ENOUGH LETTERS TO INSURE UNIQUENESS.

>>>>>DEV

IS ONE OF THE FOLLOWING ANALOG DEVICES OR VARIABLES
DIALS 0-9 (00-09)
SLIDE POTS (S0 - S9)
TEN TURN POTS (P0 - P3)
TABLET (TX, TY AND TZ)
JOYSTICKS (JX, JY, JZ) AND (KX, KY, KZ)
VARIABLES (A-Z)
(VA - VZ)
(WA - WZ)
(FA - FZ)
(SA-SH) SINE VARS
(CA-CH) COSINE VARS
(LA-LZ) LOCAL FIXED VARS
(EA-EZ) LOCAL FLOATING VARS
(OA-OH) ANALOG OUTS
PLUS THE TIME-BASED VARIABLES

SEE HELP DEV FOR HOW TO USE VARIABLES.

>>>>>VAR

IS A DEV THAT DOES NOT OBEY DEVICE CONVENTIONS AS OUTLINED IN HELP DEV. VARS ARE USED WHEN SOMETHING HAS TO BE RETURNED IN A VARIABLE
FULL-BLOWN DEV'S FOR SOME REASON.

EXPR IS A MIX OF NUMBERS, DEVS AND ARITHMETIC OPERATORS WHICH ALWAYS EVALUATES TO A SINGLE NUMBER.

EXAMPLES:

```
A
D7
200
A+(D7/200)*35-B-(K/17)
```

THERE IS OPERATOR PRECEDENCE WHEN EXPRESSIONS ARE EVALUATED I.E., EXPONENTIATION IS DONE FIRST (ONLY FOR USE WITH FLOATING POINT NUMBERS), MULTIPLICATION AND DIVISION ARE DONE NEXT, FINALLY ADDITION AND SUBTRACTION ARE PREFORMED.

PARENTHESIS CAN BE INSERTED AROUND PARTS OF AN EXPRESSION SO THAT DESIRED OPERATIONS WILL BE PERFORMED FIRST. THUS THE FOLLOWING EXPRESSION

```
(52+8)/2*12
```

EQUALS 42.

THERE IS A MOD FUNCTION "%" SO 10%4 EVALUATES TO 2.

SEE HELP FORTRAN FOR MORE DETAILS ON ARITHMETIC STATEMENTS.

$VAR IS A STRING VARIABLE AND HOLDS TEXT.

STRING MANIPULATION IS COMPLICATED EXCEPT FOR THIS CASE:

```
FROM "WHAT'S THE PIX NAME"
INPUT $A
GETOSK $A
ROATE $A, X, D0, D1
```

THESE EXAMPLE USES $A AS A DUMMY SO MACROS CAN BE GENERAL PURPOSE. SEE HELP FORTRAN FOR MORE DETAILS.

DOCUMENTATION CONCERNING SYNTAX FOR A PARTICULAR COMMAND MAY BE RETRIEVED BY TYPING HELP FOLLOWED BY THE COMMAND NAME:

```
HELP ROTATE
```

THERE ARE ALSO SYSTEM MACROS FOR MOST COMMANDS SO IF YOU DO NOT UNDERSTAND THE DOCUMENTATION IN HELP, A MACRO WILL STEP YOU THROUGH MOST COMMANDS. CALL CLIP, FOR INSTANCE, WILL RUN YOU THROUGH THE CLIP COMMAND.

IF YOU GET AN ERROR, TYPE A "?” FOLLOWED BY A CARRIAGE RETURN (AGAIN, REFERRED TO AS <CR>). THE ERROR MESSAGE WILL BE PRINTED OUT.

COMMANDS MARKED "(STRING MANIPULATION)" AND " ESOTERIC// ///" ARE BEST SKIPPED BY NOVICES.

IF THE HELP DOCUMENTATION DOES NOT MAKE SENSE, PLEASE MAKE A LOT OF NOISE ABOUT IT. IT'S HARD TO GUESS WHAT YOU DON'T UNDERSTAND.

THE HELP DOCUMENTATION IS MOSTLY FOR SYNTAX QUESTION. SOME ADDITIONAL CLUES ON HOW TO WRITE GRASS PROGRAMS ARE CONTAINED IN HELP FORTRAN, HELP DEV, HELP PIX, AND HELP MACROS.

NOTE THAT THE CNTRL W BUTTON WILL HALT A HELP PRINTOUT UNTIL YOU HIT CNTRL Q. YOU TYPE CNTRL Q AND THE OTHER CONTROL FUNCTIONS BY HOLDING DOWN THE CTRL BUTTON AND TYPING THE LETTER KEY. IT WORKS JUST LIKE A SHIFT KEY ON A TYPEWRITER.

THE CNTRL S BUTTON WILL HALT EXECUTION OF A MACRO AND PUT THE USER IN THE #-SIGN MODE. IN THIS MODE THE USER CAN PERFORM ANY OF THE SYSTEM COMMANDS. TO RETURN TO MACRO EXECUTION, TYPE ‘RESUME’.

REMEMBER TO USE THE CNTRL C BUTTON IF THINGS GET OUT OF HAND, AND THAT THE WORST YOU CAN DO IS STALL THE COMPUTER.

```
// ******************************************************************
// THERE IS NO DESCRIPTION AVAILABLE FOR THIS COMMAND AT THIS TIME, OR COMMAND HAS BEEN MISSPELLED. ------SORRY.
// ****************************************************************** IF IF
```

SYNTAX: IF VAR OPR EXPR, COMMAND

```
OR IF FS# OPR EXPR, COMMAND
CR IF $VAR OPR ALPHA-EXPR, COMMAND
```

IF IS THE SYSTEM'S CONDITIONAL FOR USE WITHIN MACROS. THE CRYPTIC SYNTAX ABOVE IS BECAUSE THE IF STATEMENT IS SO FLEXIBLE. DEV AND EXPR ARE DEFINED IN "HELP HELP"

OPR CAN BE EQ, NE, LE, LT, GE, GT, =, OR # (NOT EQUALS)

```
FS# IS FS0 - FS15 FOR FUNCTION SWITCHES (0=OFF, 1=ON)
```

$VAR IS A - Z

ALPHA-EXPR IS EITHER $A - $Z OR 'ANY STRING IN QUOTES'. IF ONE OF THE CHARACTERS IS A " " , THEN ALL STRINGS MATCHING THE CHARACTERS UP TO THE NEXT SPACE SATISFY THE FS CONDITIONS.
EXAMPLES:

IF FS1=1, DO PRETTY
IF FS1 EQ 1, DO PRETTY  
(SAME THING AGAIN)
IF DF LT 9/22, A=A+900
IF QA GT 0, SK 0
IF TX GT 0, IF TY GT 0, PUTP TX, TY, 0
IF $A=$B, SKIP 57
IF $A='YES', SKIP $YES
IF $A
'YES', SKIP $NO
IF $A='TIGER', IFN FS1; DO TIGMAC

NOTE THAT IF FA+32 GT 8+10 IS ILLEGAL, YOU HAVE TO CODE IT LIKE:
IF FA GT 8+10-32, .... ONLY VARS ARE ALLOWED ON THE
LEFT SIDE OF THE OPR.

// ************************************************************ INPUT
// SYNTAX: INPUT VAR
// OR INPUT $VAR

// INPUT IS THE COMMAND YOU USE IN A MACRO TO GET A
// RESPONSE FROM THE TELETY. YOU HAVE THE CHOICE OF INPUTTING
// NUMBERS INTO FIXED OR FLOATING VARIABLES, OR INPUTTING NAMES
// OR OTHER CHARACTER STRINGS INTO $VARIABLES.
// INPUT CAN BE FAIRLY ESOTERIC IN USE. FOR EXAMPLE, WHEN
// USING THE ARGUMENT FEATURE OF DO, CALL OR EXECUTE, THE
// ARGUMENTS ARE AUTOMATICALLY FED VIA THE INPUT COMMAND INTO THE
// MACRO. THIS MEANS THAT YOU CAN TEST THE MACRO INTERACTIVELY AND
// THEN USE IT AS PART OF ANOTHER MACRO WITHOUT REWRITING
// ANYTHING EXCEPT THE DO, CALL OR EXECUTE COMMAND LINE.
// INPUT IS NORMALLY PREFACED WITH A PROMPT SO THE USER KNOWS
// WHAT IS EXPECTED.
// INPUT/0 AND INPUT/1 THROUGH INPUT/9 ARE ESOTERIC TOO.
// INPUT/0 WILL GET THE LAST CHARACTER TYPED AND WILL NOT WAIT. IN THIS
// MODE, INPUT OFF THE KEYBOARD MAY BE USED LIKE FUNCTION SWITCHES.
// INPUT/1 WILL WAIT FOR ONE CHARACTER TO BE TYPED AND THEN
// GRAB CONTROL BACK WITHOUT WAITING FOR A <CR>.
// INPUT/5 WILL WAIT FOR FIVE CHARACTERS, ETC. NORMAL INPUT COMMAND
// USAGE ALWAYS RESULTS IN THE SYSTEM WAITING FOR A <CR>.
// INPUT OF NUMBERS MAY BE SEPARATED BY COMMAS IN BOTH
// THE INPUT COMMAND AND/OR THE RESPONSE. IF ENOUGH INFORMATION
// IS SUPPLIED, THE MACRO CONTINUES; OTHERWISE THE SYSTEM WAITS.
// FOR EXAMPLE:
// PROMPT: <PROM "ENTER THE TWO NUMBERS"
// INPUT A,B
// PROM "THE SUM IS ", A+B>

// SWITCHES:
// ALL THESE SWITCHES ARE ESOTERIC/////
// /L - TAKES ARGUMENT AS PASSED FROM DO AND THEN
// FLUSHES THE REST. (USED TO PREVENT CASE OF
// USER PASSING TO MANY ARGUMENTS.)
// /0 - INPUT VALUE IN OCTAL (BASE EIGHT)
// /T - TERMINATES INPUT FROM ARGUMENT LIST AND
// SAYS "TAKE NEXT INPUT FROM TTY AND TURN
// PROMPTS BACK ON" AS EXPLAINED ABOVE

EXAMPLES:

INPUT A  
- WHERE A IS A NUMERIC
INPUT $B  
- WHERE $B IS A CHARACTER STRING
INPUT.FC  
- WHERE FC IS A FLOATING POINT NUMBER

<<<PROMPT "WHAT PICTURE?"
+INPUT $A
+ROTATE $A, X, DO>

// **********************************************************************
// YOU ARE OUT OF LUCK. THERE IS NO HELP ON THAT COMMAND. CHECK YOUR
// SPELLING, BY TRYING IT AGAIN, OK??
// **********************************************************************
// **********************************************************************
// JUMPING JOSAPHATS---- THERE AIN'T NO COMMANDS THAT START WITH A "J".
// CHECK YOUR SPELLING.
// **********************************************************************
// **********************************************************************
// KEEP

// SYNTAX: KEEP COMMAND
// KEEP IS USED TO SAVE CERTAIN COMMANDS IN CORE IF YOU USE THEM
// OFTEN. NORMALLY, THE COMMAND IS BROUGHT IN FROM THE DISK AND
// EXECUTED, AND THEN DELETED. IF SUCH A COMMAND IS IN A LOOP,
THE FOLLOWING COMMANDS, KEEP IT. AFTER YOU'RE DONE WITH IT, USE DELETE/C COMMAND TO GET RID OF IT IF YOU'RE SHORT ON CORE.

THE COMMANDS WHICH ARE DISK RESIDENT CURRENTLY ARE:

ARRAY BACKUP BUMP COMPIL COPY CHANGE CLIP CONVRT
DIRDSK DIRCOR DIRTAP FIX HELP LOGIN LOOK MDIRC
PUTDSK PUTTEX PUSHVA POPVAR PRINT PERSP RENAME RESLEV
SEARCH SHADE SMOOTH SCFTRO SOFTEX TEXT
TREE TYPE WINDOW ZAPTEX

COMMANDS THAT KEEP THEMSELVES AUTOMATICALLY ARE

EDIT PATHMO TRACE

EXAMPLES:

KEEP GETLIB
KEEP TREE
KEEP PUTDSK

BY THE WAY, YOU HAVE TO USE THE KEEP COMMAND BEFORE YOU USE THE COMMAND ITSELF IF YOU WANT IT TO STAY IN CORE.

// ********************************************************************
*/
// THERE IS NO HELP ON THAT ONE, IT MUST BE A DOOZY. CHECK YOUR SPELLING
// BY TRYING THAT ONE AGAIN........
*/
// ********************************************************************

LENTH

SYNTAX: LENGTH CHAR STRING, VAR (STRING MANIPULATION)

LENGTH PLACES THE LENGTH OF THE STRING INTO THE VARIABLE
SPECIFIED.

EXAMPLE: *$B='GRAPHICS SYMBIOSIS'
*LENGTH $B,A
*PROMPT A
18

// ********************************************************************

LINES

SYNTAX: LINES LNAME

LINES PUTS LNAME'S VECTORS IN REGULAR DRAWING MODE. IT
IS USED TO RECOVER FROM DASHES AND POINTS MODES.

EXAMPLE:

LINES DIAMON

// ********************************************************************

LIST

SYNTAX: LIST

LIST CAUSES LINES OF A MACRO TO BE ECHOED DURING EXECUTION.
IT IS A USEFUL DEBUGGING TOOL. LIST IS TURNED OFF BY XLIST.

// ********************************************************************

LOGIN

IF YOU DON'T KNOW HOW TO USE LOGIN BY NOW, YOU'RE A SUSPICIOUS PERSON.
IN FACT, THE POLICE ARE COMING RIGHT NOW........

// ********************************************************************

LOOK

SYNTAX: LOOK ANAME, VAR

LOOK CHECKS TO SEE THE ANAME IS IN CORE
(OR WITH /D, ON THE DISK). IF IT IS ON IN CORE (OR ON DISK
WITH /D) IT RETURNS A VALUE OF 0 IN THE VAR INDICATED. IF IT IS
NOT THERE, IT RETURNS A VALUE OF 1.

EXAMPLE:

PROM "WHAT PIX"
INP $B
LOOK/D $B,DEC,C
IF C=1,PROM "$B," IS NOT ON DISK";SK -3

// ********************************************************************

LPNAME

SYNTAX: LPNAME $VAR

LPNAME RETUNS THE NAME OF THE PICTURE CORRESPONDING
TO A LIGHT PEN "HIT" IN THE $VAR INDICATED. IF NO LIGHT PEN
HIT IS SEEN, THE $VAR REMAINS UNCHANGED.

EXAMPLE:
IF $C=tI,
SKIP -1  (WAIT FOR HIT)
DELETE $C  (DELETE THE PICTURE, FOR EXAMPLE)

I LOOKED FOR A HELP LISTING FOR THAT COMMAND, BUT I COULDN'T FIND
ANY MUST NOT BE THERE RIGHT NOW, OR YOU DIDN'T SPELL IT RIGHT.

MDIRC

SYNTAX:

MDIRC IS LIKE DIRCOR EXCEPT IT DOESN'T PRINT OUT ALL THE
INFORMATION THAT DIRCOR SUPPLIES (I.E. NOTES OR TOTALS). MDIRC
TAKES UP LESS SPACE THAN DIRCOR SO IT IS USEFUL IF YOU GET
AN ERROR #13 (OUT OF CORE) WHEN TRYING TO USE DIRCOR.

MDIR

SYNTAX:

MODIFY

MODIFY ALLOWS THE USER TO LOOK AT CORE LOCATIONS FOR
DEBUGGING PURPOSES. THE OUTPUT RADIX IS SET BY SWITCHES AND
THE SYSTEM THEN WAITS FOR INPUT TO CHANGE THE CORE LOCATION.
IF A <CR> IS SEEN, NO MODIFICATIONS ARE DONE AND MODIFY IS FINISHED.
IF A <LF> IS SEEN, THE NEXT LOCATION WILL BE ACCESSED.
IF A "I" IS SEEN, THE PREVIOUS LOCATION WILL BE ACCESSED.
IF A "@" IS SEEN, THE CONTENTS OF THE ADDR IS USED AS AN ADDR (INDIRECT).
IF A NUMBER IS SEEN, IT WILL REPLACE THE PRESENT LOCATION'S CONTENTS
WITH THAT NUMBER AND DISPLAY THE NEXT LOCATION.
IF EXPRl,VAR,EXPR2 ARE USED: EXPRl IS ADDED TO ADDR(DECIMAL)
VAR GETS OLD CONTENTS OF LOCATION
EXPR2 REPLACES LOCATION IN CCRE(DECIMAL)
IF EXPR IS LEFT OUT ENTIRELY, MOD USES THE LAST ADDRESS USED BY
MOD OR SETBFK.

SWITCHES:

/A - FOR ASCII REPRESENTATION
/B - FOR BYTE REPRESENTATION
/D - FOR DECIMAL REPRESENTATION
/O - FOR OCTAL REPRESENTATION (DEFAULT)
/V - FOR VECTOR GENERAL ADDRESSES

EXAMPLES:

MOD VG+2000
MOD GRAPE+100
MOD 77646

MOVE

SYNTAX:

MOVE PNAME,DEVl,DEV2,DEV3

THE MOVE COMMAND TRANSLATES THE PNAME ALONG THE COORDINATE
AXES THE AMOUNT INDICATED BY DEVl FOR X, DEV2 FOR Y AND DEV3 FOR Z.
MOVE USES THE DEVICE CONVENTIONS OUTLINED IN "HELP DEV"

EXAMPLE:

MOVE GRAPE,DI
MOVE TOM,F
THE FIRST EXAMPLE MOVES THE GRAPE ACCORDING TO D1, D2, AND D3.
THE SECOND EXAMPLE MOVES TOM ACCORDING TO THE VARIABLES F,G, AND H.
MOVE SAM,D5,RA,1000
THIS MOVES SAM ACCORDING TO D5 (X-AXIS), VARIABLE RA(Y-AXIS)
AND A CONSTANT 1000 (FOR Z-AXIS).

HMMMMM, SOMETHING IS AMISS HERE, EITHER THERE ISN'T ANY HELP ON
THAT COMMAND, OR YOU CAN'T SPELL IT RIGHT. LET'S TRY IT ONE MORE
TIME, OK?

NO NO NO, YOU'RE GONNA GET NO WHERE FAST ASKING FOR HELP ON A COMMAND
THAT STARTS WITH AN "H". LET'S FACE IT, YOU BLEW THE SPELLING

ONERPC
ONERROR SETS UP A COMMAND TO BE EXECUTED IN THE EVENT AN ERROR OCCURS. THE SYSTEM STICKS THE ERROR NUMBER IN THE VARIABLE INDICATED AND EXECUTES THE COMMAND YOU SPECIFIED AFTER THE COMMA IN PLACE OF THE COMMAND WHICH CAUSED THE ERROR. (NOTE THAT IT ALSO THROWS OUT THE REST OF THE LINE IN ERROR TOO.)

SKIPS IN ONERROR COMMANDS ARE TAKEN RELATIVE TO THE LOCATION OF THE COMMAND IN ERROR, NOT TO THE LOCATION OF THE ONERROR COMMAND ITSELF. IF THIS CAUSES CONFUSION, REMEMBER TO HAVE ALL ONERROR SKIPS REFERENCE %LABELS INSTEAD. THEN IT'S EASY AND EVEN EASY TO FIGURE OUT. YOU CAN EVEN CALL A MACRO IN THE EVENT OF AN ERROR. THIS IS IMPORTANT IF THERE IS A POSSIBILITY OF MULTIPLE ERRORS OCCURRING. THIS IS WHY THE ERROR NUMBER IS PUT IN THE VARIABLE.

EXAMPLE:

```
ONERROR VA,DO FIXUP
WHERE FIXUP IS:
  FIXUP: <IF VA=13,DELETE $B;RETURN
           IF A=1,GETDSK $A;RETURN
           ..ETC.>
```

THE ERROR NUMBERS ARE GIVEN IN ERRMES.SYS 31,4 IN THE RIGHT MARGIN.

SWITCHES: /A - IGNORE ALL ERRORS

NOTE: ONERROR WITHOUT ANY OPERAND CANCELS THE SETUP. YOU SHOULD CANCEL ONERRORS WHEN YOU ARE DONE WITH THEM SO THEY DO NOT DO WEIRD THINGS WHEN AN ERROR IS ENCOUNTERED SOME TIME LATER ON. RESTART CANCELS ALL ONERRORS.

EXAMPLE:

```
ONERROR A, GETDSK WITCH, DEC 31,1
GETDSK WITCH
  IN THIS EXAMPLE, IF THE WITCH IS NOT FOUND IN THE USERS AREA, INSTEAD OF GIVING AN ERROR MESSAGE, THE WITCH WILL BE TAKEN FROM THE COMMON AREA OF THE DISK.
```

SYNTAX: OPEN PIX

OPEN IS USED TO SET THE NAME FOR FUTURE PUTPOI'S AND DELPOI'S. IF YOU DO A PUTPOI WITHOUT AN OPEN, YOU WILL GET AN ERROR MESSAGE.

EXAMPLES:

```
OPEN SAM
OPEN $A
```

THE NORMAL SEQUENCE IS:

```
OPEN PIX
X=1000
PUTPOI X,X,0,0
PUTP X,-X,0,0
PUTP -X,-X,0,0
PUTP -X,X,0,0
PUTP X,X,0,0
CLOSE
```

THIS WILL DRAW A BOX.

A SIMPLE MACRO TO COPY EVERY OTHER POINT IN A PICTURE WOULD BE:

```
<OPEN FOO
N=0
%MORE N=N+1
GETP OLD,N,X,Y,Z,K
IF K=-1,CLOSE;RETURN
N=N+1
GETP OLD,N,X,Y,Z,K
IF K=1,PUTP X,Y,Z,K;SK %MORE
CLOSE>
```

NOTE THAT K=1 IN A PUTPOI IS REALLY TAKEN AS A 0. IT DOES NOT TERMINATE THE PICTURE AS IT WOULD IN ZAPPOI. ONLY CLOSE CAN TERMINATE AN OPENED PICTURE.

```
// ************************************************************************
// AH-ON, I CAN'T SEEM TO FIND HELP ON THAT COMMAND. IT MIGHT NOT BE HERE, BUT CHECK YOUR SPELLING, AND TRY THAT ONE AGAIN.
// ************************************************************************
```

SYNTAX: PATHMOV
PATHMOV moves PNAME tangentially along PIX. The
VAR indicates the position along the path and ranges from 0 to the
number of points in PIX. Resetting the VAR anytime causes
the PATHMOV to resume at that point. Setting it to zero starts it
from the beginning.
The first DEV is the speed. Below 0, it
causes an interpolation between points. Full negative stops it.
If the DEV is positive, the speed is taken as that many
points per 1/60 second. The second (optional) DEV is used to
give fineness which helps smooth the path somewhat.
PATHMOV does not use the new DEV conventions in HELP DEV.
When the end of the path is reached, the VAR is set to -1.

EXAMPLE:
PATH BIRD, PAT1, A, DO
DOL <IF A LT 0, A = 0>

SEE HELP DOLOOP IF THIS IS CONFUSING. BASICALLY, IT
JUST SETS A TO ZERO WHENEVER IT GOES NEGATIVE.

PATHOFF

SYNTAX:
PATHOFF

PATHOFF turns the light pen off.

PERSP

SYNTAX:
PERSP PIX1, PIX2, AEXPR, SCALE
WHERE SCALE IS AN EXPR

PERSP creates a new picture by making various hardware
modifications to a picture (specified by PIX1), and putting the
modified picture into the name given by PIX2.
PERSP then does a viewing perspective transformation on
the new picture, by assuming that the viewer is at a screen coordinate
specified by AEXPR on the Z-axis. 500 is the default value used
for the SCALE argument. A scale factor of 1000 will double the
size of the output picture. 250 will reduce the size of the
output picture by 1/2, etc.
The old picture is not affected in any way by doing a
perspective.

SWITCHES:
/C - OPERATES THE SAME AS ABOVE EXCEPT HARDWARE
MODIFICATIONS ARE IGNORED AND AEXPR CAN
BE SUBSTITUTED WITH XEXPR, YEXPR, ZEXPR
THAT WILL SPECIFY THE POINT OF PERSPECTIVE
CRGIN (i.e. THE /C SWITCH ASSUMES THE
VIEWER IS LOOKING TOWARD THE SCREEN CRGIN
FROM THE COORDINATES GIVEN BY XEXPR, YEXPR
AND ZEXPR, ONE OF WHICH MUST BE NON-ZERO)

EXAMPLES:
PERSP TV, FOO, 500, 250
THIS WILL DO A PERSPECTIVE TRANS. ON THE PICTURE
TV WITH THE VIEWER AT Z-COORDINATE 500. THE
RESULTING PICTURE IN "FOO" WILL BE 1/2 THE SIZE
OF THE ORIGINAL.
PERSP/C STEEPLE, DOW, 0, 2000, 1000, 1000
THIS PERSPECTIVE WILL BE DONE BY PLACING THE
VIEWER AT COORDINATES 0, 2000, 1000 AND THE
RESULTING PICTURE WILL BE TWICE NORMAL SIZE.
/C PERSPS ARE GOOD FOR FOLLOWING A PATH THROUGH AN OBJECT.
USE THE PATH COORDINATES TIMES 16 AS ARGUMENTS TO PERSP/C.

POINTS

SYNTAX:
POINTS PIX

POINTS changes PIX's vectors to display only the endpoints.

EXAMPLE:
POINTS MOREY

POINTS IS THE OPPOSITE OF PUSH. IT IS USED TO RESTORE PUSHED
VARIABLES. THE ARGUMENTS ARE PAIRED AND MUST BE IN THE
ORDER A-Z, VA-VZ, WA-WZ, & FA-FZ. THE ARGUMENTS SPECIFIED ARE
NOT RESTORED TO THEIR OLD VALUES.
EXAMPLES:

POP C,FZ
POP A,M,O,VB,VE,FK,FM,FZ
POP VA,VA

RESTORES ALL BUT A & B
RESTORES ALL BUT N, VC
VD, & FL
POPS JUST VA

// ************************************************************* PRINT
PRINT

SYNTAX:

PRINT DNAME, EXT XX,XX

PRINT IS THE SAME AS "TYPE" BUT IT CREATES HARDCOPY ON THE
LINE PRINTER INSTEAD OF DISPLAY ON THE VT05. ANYTHING AFTER
THE NAME (PLUS SPACE) IS TREATED AS A COMMENT AND IS PRINTED AT
THE TOP OF THE PAGE. DEFAULT PRINTING IS SINGLE SPACING.

SWITCHES:

/O - THE HARDCOPY IS PRINTED WITH DOUBLE
SPACING (WASTES PAPER).
/N - SUPPRESS THE DATE AND TIME

EXAMPLE:

PRINT DRAW.MAC
PRINT PATH.DEC 50,53

// ************************************************************* PROMPT
PROMPT

SYNTAX:

PROMPT THING,THING,THING......

PROMPT IS THE COMMAND USED TO OUTPUT VALUES OF VARIABLES OR
DEV'S TO THE VT05. IT CAN ALSO BE USED TO
WRITE MESSAGES TO A MACRO USER AND THUS PROVIDES A MEANS FOR
ASKING QUESTIONS BEFORE INPUT COMMANDS, TYPING INFORMATION OUT, ETC.
THING IS EITHER A VARIABLE (STRING OR ARITHMETIC) OR A BUNCH OF LETTERS
IN DOUBLE QUOTATION MARKS. IF A "|" IS THE LAST CHARACTER ON THE
LINE, THE CARRIAGE RETURN WILL BE SUPPRESSED AND THE INPUT CAN
THEN APPEAR ON THE SAME LINE AS THE ONE TYPED OUT. IF ANY ARGUMENTS ARE
PASSED TO THE MACRO ALL PROMPTS WILL NOT PRINT TO THE TERMINAL UNTIL
ALL OF THE ARGUMENTS ARE GONE OR INPUT/T IS USED.

SWITCHES:

/F - FORCES PROMPT EVEN IF SUPPRESSED BY
ARGUMENTS IN MACRO CALL.
/L - PROMPTS ON THE LINE PRINTER.
/P - IS USED TO ACTUALLY PRINT THE DATA
PREVIOUSLY SENT TO THE LINE PRINTER.
NOTE: THIS HAS NO ARGUMENTS.
/O - PRINT OCTAL VALUE OF VARIABLE

EXAMPLES:

PROMPT DO
PROMPT "THE ANSWER IS ",D0
PROMPT "A+B="",A+B,"AND C+D= ",C+D
PROMPT "PUSH FS13 TO EXIT"
PROMPT "X COORDINATE"
PROMPT FA,FB,C

PROM "WHAT PIX DO YOU WANT TO SEE"|
INP $A
GETDSK $A

// ************************************************************* PUSHVA

PUSHVA

SYNTAX:

PUSHVAR

PUSH PUSHES ALL THE VARIABLES IN THE ORDER A-Z, VA-VZ, WA-WZ,
& FA-FZ IN A PUSHDOWN STACK MANNER (20 DEEP).
EACH UTILITY MACRO SHOULD PUSH UPON ENTRY & POP UPON EXIT
TO AVOID CONTAMINATING THE USER'S VARIABLES. IT IS THE MACRO WRITERS
RESPONSIBILITY TO PAIR THE PUSHES & POOPS CORRECTLY.

SWITCHES:

/C - CLEARS THE VARIABLES AFTER PUSHING THEM
(I.E. SETS THEM TO 0).
NOTE: UNLESS THIS SWITCH IS USED, THE VARIABLES
ARE NOT CHANGED BY PUSH SO ARGUMENTS MAY STILL
BE PASSED IN VARIABLES.

NOTE: PUSH AND POP WORK POORLY WITH DOLOOPING.

// ************************************************************* PUTDSK

PUTDSK
PUTDSK PUTS DNAME OUT TO YOUR AREA ON THE DISK IN THE FORMAT
INDICATED BY THE EXTENSION.

EXTENSIONS:
• DEC  PICTURES(DEFAULT)
• MAC  MACROS
• CPL  COMPILk FILES
• ARR  ARRAY FILES

WHEN THE EXTENSION IS NOT ONE OF THE ABOVE, THE FILE
WILL BE PUT ON DISK IN .MAC FORMAT, EVEN THOUGH IT MAY HAVE
ANOTHER THREE LETTER EXTENSION. THERE IS ALSO AN OPTIONAL
ARGUMENT WHICH MUST BE A $VAR. IF THIS IS SPECIFIED, THE $VAR'S
CONTENTS WILL BE PUT ON THE DISK WITH THE DNAME SPECIFIED.
IS THE EXT IS NOT PRESENT PUTDSK WILL TRY TO FIGURE OUT WHAT IT IS AND
USE THE PROPER EXTENSION FROM ABOVE. IF NO EXT IS SPECIFIED
THE PROPER ONE WILL BE ASSUMED.

YOU CANNOT PUTDSK ON ANY AREA BUT YOUR OWN.

SWITCHES:
/D - CREATES A BACKUP FILE WITH THE EXT .BAK IF
    THE FILE ALREADY EXISTS ON THE DISK. VERY
    USEFUL WITH THE QUICKY EDITOR (EDIT).
/A - THIS WILL ADD TO THE FILE ALREADY ON DISK.
    THIS COULD CREATE VERY LARGE DISK FILES IF
    USED INDiscrimINATELY. IF THE DISK FILE DOES
    NOT EXIST. THE /A IS IGNORED AND THE COMMAND
    PROCEEDS AS A NORMAL PUTDSK.
/M - COMPLEMENT TO GETDSK/M, AS IT IgNORES
    EXTENSION CHECKING. IT PUTS THE FILE
    ON THE DISK AS IF IT WERE A MACRO.
/B - COMBINATION OF /M & /A

EXAMPLES:
PUTDSK DRAW.MAC
PUTDSK GLOBE
PUTDSK WITCH.DEC
PUTDSK TEMP.MAC,$C
PUTDSK HELLC.TRY

PUTLIB

SYNTAX:
PUTLIB PNAME

PUTS PNAME INTO NON-DISPLAYED DATA STRUCTURE AND REMOVES
PNAME FROM HIGHER GROUPS IF ANY. PNAME MUST BE DISPLAYED.
THE COMMANDS WHICH WORK ON A PUTLIB'D PNAME ARE AS FOLLOWS:
BLANK  CUTOFF  CLOSE  COPY  DASHES  DELETE  DELPOD
FIXGETLIB  LOOK  LINES  MOVE  PATHMOV  PERSP  POINTS
PUTDSK  PUTPOI  PUTTEX  RENAME  RESET  ROTATE  SCALE  SETCO
SETINT  SETORG  SHADE  SMOOTH  SOFTRO  ZAPPOR  ZAPTEX  WINDOW
NOTE THAT THE UPDATES TO PNAME ARE NOT DONE WHILE THE PNAME IS
PUTLIB'D. TO HAVE THE UPDATES DONE BUT NOT USE UP COMPUTER TIME
TO DISPLAY THE PIX (FOR PIX ONLY) YOU CAN USE THE BLANK COMMAND.
GETLIB/W IS ALSO USEFUL FOR GETTING THE PNAME BACK
Smoother.

EXAMPLE:
PUTLIB OUTLINE

PUTPOI

SYNTAX:
PUTPOI X,Y,Z,K
WHERE X,Y,Z,K ARE EXPRs

DP
PUTPOI/C PNAME,EXPR1,EXPR2,K

PUTPOI PUTS A POINT INTO THE PIX INDICATED BY THE PREVIOUS OPEN.
X,Y,Z ARE ANY EXPRESSIONS DETERMINING THE X,Y, AND Z COORDINATES OF THE
POINT IN DECIMAL (RANGE -2048 TO 2047). K INDICATES THE FOLLOWING:
K=0 DRAW THE VECTOR
K=1 MOVE TO NEW LOCATION BUT DO NOT DRAW (THAT IS, JUMP)
TO TERMINATE THE LIST, USE THE CLOSE COMMAND. ALSO NOTE THAT
THE FIRST POINT IS ALWAYS STORED AS K=1.
SEE HELP PIX FOR MORE INFORMATION ON CREATING PICTURE LISTS.

SWITCH: /C

EXAMPLE:
PUTPOI 0,0,0,1

COPIES A RANGE OF POINTS FROM ANOTHER PIX

EXAMPLE:
PUTPOI 0,0,0,1

THIS WILL CREATE A PICTURE WITH A LINE DRAWN FROM THE
INSTANTANEOUS POSITIONS OF DO,DI,AND D2 TO THE POSITIONS
OF THE VARIABLES A, B, AND C. (THE /16 IS NECESSARY BECAUSE DIALS
ARE READ LEFT JUSTIFIED.)
EXAMPLE:
OPEN SAMBOX

PUTPOI 0.0,0.0,0.0
THIS WILL CREATE A BOX IN THE UPPER RIGHT QUADRANT OF THE SCREEN. THE BOX IS NAMED SAMBOX.

EXAMPLE:
OPEN BEEP
PUTP/C PAUL,10,20,1

THIS WILL COPY VECTORS 10 TO 20 OF PAUL INTO BEEP, STARTING OFF WITH AN UNDRAWN VECTOR. REGULAR PUTP/C'S CAN PRECEED AND FOLLOW PUTP/C'S.

// ********************************************** PUTTEX

PUTTEXT SYNTAX:

PUTTEXT XEXPR, YEXPR, LEXPR

+ SOME TEXT TO BE USED


SWITCHES:
JUST LIKE IN TEXT

NOTE:
TEXT SHOULD BE PUT AT THE END OF THE PICTURE IF THE PICTURE IS TO LATER BE MANIPULATED WITH GETPOI OR ZAPPOI. DON'T PUT A "*" AS THE FIRST CHARACTER, BECAUSE THE SYSTEM WILL THINK IT'S A COMMENT.

// *************************************************

SORRY ABOUT THIS, BUT THE INFORMATION THAT YOU WANT DOESN'T EXIST AT THIS POINT IN TIME. IS YOUR SPELLING INOPERATIVE??

// *************************************************

QUIT IT WILL YA, YOU MUST BE QUAZY IT THE HEAD. YOU TRIED TO GET HELP ON A COMMAND THAT STARTS WITH A "Q". CHECK YOU'RE SPELLING.

// *************************************************

RENAMF RENAME

SYNTAX:
RENAME ANAME1, ANAME2

RENAME SIMPLY CHANGES THE FIRST NAME TO THE SECOND NAME.

SWITCHES:
NONE AS ABOVE
/D RENAMF DISK FILE ON YOUR AREA

EXAMPLES:
GETDSK GRAPE
RENAME GRAPE, PRUNE
GETDSK GRAPE

NOW THERE ARE TWO COPIES OF GRAPE ON THE VG SCREEN.

RENAME/D GRAPE.DE, PRUNE.DE
GRAPE.DE IS NOW CALLED PRUNE.DE ON THE DISK.

// *************************************************

RESET

RESET SYNTAX:
RESET PNAME

RESET IS THE SAME AS FIX EXCEPT THAT THE ORIGINAL VALUES OF THE ROTATION MATRIX, ORIGIN, SCALE, INTENSITY, ETC. ARE RESTORED.

SWITCHES:
SAME AS FIX

EXAMPLE:
RESET/R PROP

// *************************************************

RESOLVE

SYNTAX:
RESOLVE MNAME1, MNAME2

RESOLVE CONVERTS %LABEL PSEUDO-LABELS IN MNAME1. IF A MACRO HAS PSEUDO-LABELS, RESOLVE CAN BE USED TO REMOVE THEM. THE ONLY REASON TO USE IT IS TO MAKE THE MACRO RUN SLIGHTLY FASTER. PSEUDO-LABELS ARE USED TO ELIMINATE THE TASK OF COUNTING LINES FOR SKIP ARGUMENTS IN LONG MACROS. MNAME1 IS AUTO-DELETED. IF THE SINGLE MNAME IS USED THE RESOLVED MACRO WILL HAVE THE SAME NAME AS THE UNRESOLVED ONE.

NOTE: WHEN YOU ORIGINALLY DEFINE THE %LABEL YOU MUST HAVE A SPACE OR A TAB IMMEDIATELY FOLLOWING THE LABEL.
EXAMPLE:
GLOB:<IF FS1, SKIP %FLUFF
SKIP -1
%FLUFF DIRCOR>
RESOLVE GLOB, CGLOB
CGLOB NOW LOOKS LIKE:
CGLOB:<IF FS1, SKIP 2
SKIP -1
DIRCOR>
GLOB IS AUTO-DELETED AND CGLOB CAN BE EXECUTED.
NOTE THAT THIS ALL REALLY ISN'T NECESSARY SO FORGET RESOLVE.

// **************************** RESUME
RESUME

SYNTAX: RESUME

RESUME IS HOW YOU GET BACK TO THE MACRO IN PROGRESS WHEN
YOU ARE IN #-MODE. YOU GET INTO #-MODE FROM TYPING [S, WHEN
AN ERROR CONDITION IS ENCOUNTERED, OR WHEN YOUR MACRO
HAS A WAIT COMMAND EXECUTED (FAIRLY RARE, ACTUALLY). YOU
CAN ALSO GET INTO #-MODE WITH A [S DURING A GETDSK.
IN #-MODE YOU CAN EXECUTE ANY GRASS COMMANDS OR
EVEN MACROS BUT ALL FURTHER [S USAGE IS IGNORED UNTIL RESUME IS TYPED.

// **************************** RESTART
RESTART

SYNTAX: RESTART

RESTART RE-INITIALIZES THE ENTIRE GRASS SYSTEM.
IF THE CALL OR DO MNAME IS INCLUDED, RESTART CLEARS THE SYSTEM AND THEN
DOES THAT MACRO AUTOMATICALLY. (ACTUALLY, MOST COMMANDS WILL WORK
--E.G., RESTART GETC GRAPE-- BUT FEW ARE USEFUL IN THIS CONTEXT.)

SWITCHES: /V - THE VALUES OF THE ARITHMETIC VARIABLES
WILL NOT BE SET TO ZERO.

EXAMPLES:
REST
RESTART CALL DEMORY
RESTART DO CTITLE
RESTART DO DBLPCI.MAC XX,XX

NOTE THAT UPON RESTART, VARIABLE WZ IS RESET TO 32767.

// **************************** RETURN
RETURN

SYNTAX: RETURN

RETURN IS USED TO LEAVE A MACRO FROM THE MIDDLE. IT IS
CONCEPTUALLY AND OPERATIONALY EXACTLY EQUIVALENT TO A
SKIP 9999.

EXAMPLE:
FIXUP: <IF A=13, PROM "OUT OF CORE"; RETURN
IF A=1, PROM "CAN'T FIND THAT" RETURN
IF A=2, PROM "THAT'S DUMB">

NOTE THAT A RETURN IS ASSUMED AT THE END OF EVERY MACRO. YOU DO NOT
HAVE TO PUT ANYTHING SPECIAL AT THE END OF A MACRO; IT RETURNS AUTOMATICALLY.

// **************************** ROTATE
ROTATE

SYNTAX: ROTATE

ROTATE TAKES THE PNAME AND Rotates IT ACCORDING TO THE
AXIS (MUST BE INDICATED AS "X", "Y", OR "Z") AT THE
SPEED (OR WITH /D, THE ANGLE) INDICATED BY A DEV. IF INCLUDED,
THE TILT MODIFIES THE AXIS ACCORDING TO THE SETTING OF A DEV.
IF INCLUDED, DEV1, DEV2, AND DEV3 ARE THE ORIGIN OF THE AXIS
OF ROTATION.

THE /D SWITCH MAKES THE SPEED INTO AN ANGLE.
TO GET AN ANGLE OF 180 DEGREES, SET THE SPEED DEV TO 32767 (=WZ)
90 DEGREES, SET THE SPEED DEV TO 16383 (=WZ/2)
45 DEGREES, SET THE SPEED DEV TO 8191 (=WZ/4)
30 DEGREES, SET THE SPEED DEV TO WZ/6 ETC.

ROTATE USES THE DEVICE CONVENTIONS OUTLINED IN "HELP DEV"
SWITCHES:
/D DEV INDICATES ANGLE OF ROTATION INSTEAD OF SPEED
/X COMPLEX ROTATION ADDED TO REGULAR ROTATION (X)
NOTE: ON /X, /Y, /Z ROTATIONS, THE AXIS IS NOT SPECIFIED AGAIN.

ALSO NOTE: A REGULAR ROTATION MUST BE DONE BEFORE A /X, /Y, /Z MAY BE USED.

ALSO ALSO NOTE: SINCE ROTATIONS ARE NOT COMMUTATIVE, IT IS IMPORTANT TO KNOW THAT THE ORDER YOU SPECIFY THE /X, /Y OR /Z ROTATIONS WILL CHANGE THE OVERALL ROTATION.

IF THE CHARACTER '7' IS SPECIFIED IN PLACE OF THE AXIS OF ROTATION, SEVEN DEVS ARE EXPECTED, ONE FOR SPEED AND SIX TO INDICATE THE ENDPOINTS (X1, Y1, Z1 AND X2, Y2, Z2 IN THAT ORDER) OF THE AXIS OF ROTATION TO BE USED.

ALL SWITCHES WORK IN THE SEVEN-DIAL ROTATE (/D ETC.) TOO.

SETORG WORKS WEIRDLY WITH 7-DIAL BUT DOES A PRE-ROTATION MOVE WITH 2 OR 5 DIAL ROTATES. SETORG DOES NOTHING WITH SINGLE DIAL ROTATES EXCEPT SET THE ZOOM POINT FOR SCALES WITH SWITCHES. (SEE HELP SCALE, HELP SETORG)

EXAMPLES:

- ROAT PROP, X, D9
- ROAT PROP, X, D9
- ROAT GLOBE, Z, D8
- ROAT GLOBE, Z, D8
- ROAT BLADE, Z, D0, D1
- ROAT BLADE, Z, D0, D1
- ROAT/X BLADE, D5
- ROAT/Y BLADE, D6

(THREE HEE Instructions CAUSE A ROTATION WITH LOTS OF CONTROL)

- ROAT SAM, X, D0, D1, A
- ROAT/FRED, K, L, F
- ROAT LARRY, 7, D0, A, B, C, D, E, F WHERE ENDPOINTS OF THE AXIS OF ROTATION ARE IN VARS A-F

---

*/  *********************************************************************/

THAT INFORMATION ISN'T IN THE FILES AS OF YET, BUT CHECK YOUR SPELLING JUST TO MAKE SURE.

---

*/  *********************************************************************/

SCALE

SYNTAX:  SCALE PNAME, DEV

SCALE CHANGES THE SIZE OF PNAME BY A FACTOR INDICATED BY THE DEV. THIS SCALE ONLY SCALES DOWN FROM THE ORIGINAL SIZE. TO SCALE UP, USE THE MACRO "CALL BIGGER".

SWITCHES:

NONE USES HARDWARE SCALE WHICH SCALES TO PICTURE'S ORIGIN AS MODIFIED BY MOVE, ROTATE, ETC.
/X SCALES ALONG X AXIS (SINGLE DIMENSIONAL SCALING).
/Y SCALES ALONG Y AXIS.
/Z SCALES ALONG Z AXIS.
/A SINGLE DIMENSIONAL SCALING ON DEV, DEV+1 AND DEV+2.
/F SAME AS /A BUT ALL ON SAME DEV SCALING WITH SWITCHES ALSO ALLOWS THE ORIGIN OF SCALE TO BE RESET BY USING SETORG COMMAND.

EXAMPLES:

- SCALE TOM, D0
- SCALE/X NOSE, D9
- SCALE/A SPIRAL, J

THE LAST LINE TAKES J FOR X SCALE, K FOR Y AND L FOR Z SCALE.

NOTE: SCALE USES THE NEW DEVICE CONVENTIONS (SEE HELP DEV).

---

//  *********************************************************************/

SEARCH

SYNTAX:  SEARCH/P. SA, $C, SD

THE SEARCH COMMAND IS A STRING MANIPULATOR COMMAND WITH THE FOLLOWING SYNTAX:

SEARCH/P $A, $B, $C, $D  - IN THE TARGET STRING $A, START SEARCH AT LOCATION $B RELATIVE TO 1ST STRING POSITION, SEARCH FOR FIRST OCCURRENCE OF $C AND REPLACE WITH $D. (NULL ARG FOR B STARTS SEARCH AT POSITION 1.)

SEARCH/P $A, $B, X$C, $D  - LIKE ABOVE EXCEPT SEARCH FOR THE XTH OCCURRENCE OF $C WHERE X CAN BE ANY LEGIT GRASS ALGEBRAIC EXPRESSION.

SEARCH/P $A, $B, $C, $D  - LIKE ABOVE EXCEPT SEARCH FOR ALL OCCURRENCES OF $C AND REPLACE WITH $D.

SEARCH/P $A, $B, $C  - LIKE ABOVE EXCEPT SEARCH FOR THE STRING $A AND $B AND $C AND REPLACE WITH $D.

SEARCH/P $A, $B, $C  - LIKE ABOVE EXCEPT SEARCH FOR THE STRING $A AND $B AND $C AND REPLACE WITH $D.
SEARCH/F $A, B, X+C, D - STARTING AT LOCATION B, SEARCH $A FORWARD FOR THE XTH OCCURRENCE OF $C AND PUT FIRST CHAR'S LOCATION IN D. (DEFAULT IS THE FIRST OCCURRENCE IF X IS OMITTED).

SEARCH/L $A, B, -C, D - STARTING AT B, SEARCH $A BACKWARDS FOR THE XTH OCCURRENCE OF $C AND PUT THE FIRST MATCHING CHAR'S LOCATION IN D.


SEARCH/S $A, B, C, $D - IN $A AT B FOR C CHAR'S REPLACE WITH $D.

// ****************************************************************** SETBRK

SYNTAX: SETBRK EXPR

OR SETBRK ANAME+EXPR (INCLUDING ALL NAMES IN NAMES TABLE)

SETBRK SETS US A BREAK-POINT AT THE LOCATION INDICATED.

THE REGISTERS ARE PRINTED OUT AND THEN ANY COMMAND INCLUDING MODIFY MAY BE USED TO HELP DEBUG SOME CODE. TO DEBUG SWAP MODULES, "KEEP" THEM AND SET THE BREAKPOINT AFTER THEY ARE IN CORE. CORE RESIDENT MODULES MAY BE Debugged BY SETTING BREAKPOINTS AT ABSOLUTE LOCATIONS OR OFFSETS TO THE VARIOUS MODULES (E.G. VG+2276, SUB+5000). REMOVE THE BREAKPOINT BY TYPING SETBRK 0 OR SETTING A DIFFERENT BREAKPOINT LOCATION. YOU CAN RESUME THE CODE BY TYPING "RES", ALTHOUGH THE BREAKPOINT WILL GO AWAY.

EXAMPLES: SETBRK DIRCOR+2200

SETBRK HELPA44

// ****************************************************************** SETCO

SYNTAX: SETCO PNAME, DEV

SETCO SETS THE CUTOFF PLANE (Z-AXIS CUEING) ACCORDING TO THE VALUE SPECIFIED BY THE DEV. IF THE DEV IS SET TO SLIGHTLY NEGATIVE, THEN VARYING THE INTENSITY SET BY SETINT WILL MOVE THE CUTOFF PLANE WITH RESPECT TO THE Z-AXIS. IF DEV IS SET TO FULL POSITIVE, THE DEPTH CUEING FEATURE WILL BE MOST PROMINENT WHEN THE INTENSITY DEV IS VARIED.

EXAMPLE: SETCUT GLOBE, D8

NOTE: SETCO USES THE NEW DEV CONVENTIONS (SEE HELP DEV)

// ****************************************************************** SETDEL

SYNTAX: SETDEL EXPR

THE DELAY, INDICATED BY A EXPR, IS USED TO SLOW DOWN GETDSK FOR .DEC MODE. SETDELA REMAINS SET FOR THE REST OF YOUR SESSION OR UNTIL RESET BY, FOR EXAMPLE:

SETDELA 0 ANYTHING LARGER THAN 10 IN SETDELA IS PRETTY SLOW.

// ****************************************************************** SETINT

SYNTAX: SETINT PNAME, DEV

SETINT SETS THE INTENSITY OF THE PNAME AS INDICATED BY THE DEV USED.

EXAMPLE: SETINT GLOBE, D0

NOTE: SETINT USES THE NEW DEV CONVENTIONS (SEE HELP DEV)

// ****************************************************************** SETORG

SYNTAX: SETORG PNAME, DEV1, DEV2, DEV3

SETORG SETS THE ZOOM POINT OF A PICTURE FOR SINGLE-DIMENSIONAL SCALING. IT DOES NOT AFFECT REGULAR SCALING. SETORG ALSO SETS THE PRE-ROTATION TRANSLATION FOR 2 AND 5-DIAL ROTATES.

EXAMPLE: SETORG SAM, JS

NOTE: SETORG USES THE NEW DEV CONVENTIONS (SEE HELP DEV)

// ****************************************************************** SHADE

PIX2 IS THE NAME GIVEN TO THE GENERATED SHADE VECTORS.
SPACING IS THE DISTANCE BETWEEN THE SHADE VECTORS.
IF SPACING IS NOT GIVEN THE DEFAULT VALUE IS 10 UNITS.
ANGLE IS THE ANGLE, IN DEGREES, AT WHICH THE SHADE VECTORS ARE DRAWN. IF NOT GIVEN AN ANGLE IS CALCULATED THAT USES THE LEAST AMOUNT OF CORE SPACE.

BE CAREFUL WHEN USING SHADE. BE SURE PIX1 IS DRAWN IN PROPER FORMAT AND WATCH SPACING OF SHADE VECTORS BECAUSE CORE GETS USED UP FAST WITH CLOSE SPACING.

EXAMPLES:

SHADE PRYM,P00,20,90
SHADE BOX,SAM

// *********************************************** SKIP

SYNTAX:

OR SKIP %LABEL

SKIP IS USED TO TRANSFER CONTROL WITHIN A MACRO FOR LOOPING.
EXPRESSION MUST EVALUATE TO A NUMBER WHICH, IF NEGATIVE, SKIPS BACKWARDS THAT NUMBER OF LINES. IF POSITIVE, CONTROL SKIPS FORWARD THAT NUMBER OF LINES. NUMBERS LARGER THAN POSSIBLE WITHIN THE MACRO WILL RESULT IN TRANSFER TO THE FIRST STATEMENT OF THE PROGRAM IF NEGATIVE, OR PAST THE LAST (AND THEREFORE A RETURN) IF LARGE POSITIVE.

EXAMPLE:

<X=X+1
PROMPT X
IF FS1=0, SKIP -2>
THIS EXAMPLE WILL INCREMENT X AND PRINT IT UNTIL FS1 IS PUSHED.

<.....
IF FS2=0, SKIP 0
.....>
THIS EXAMPLE WILL WAIT UNTIL FS2 IS PUSHED BEFORE CONTINUING.

<.....
%BACK
%BACK N=N+1
IF N LT 200, SKIP %BACK
.....>
THE ABOVE SHOWS THE USE OF %LABELS. YOU SHOULD USE %LABELS NEARLY ALL THE TIME TO HELP PREVENT THE ONSET OF INSANITY.

// *********************************************** SMOOTH

SYNTAX:

SMOOTH PIX,EXPR

SMOOTH USES A MODIFIED QUADRATIC SMOOTHING TECHNIQUE TO SMOOTH THE 3-D PIX THE NUMBER OF TIMES INDICATED BY THE EXPR.

EXAMPLES:

SMOOTH PATH,7
SMOOTH GLOBE,11/8-A

// *********************************************** SOFTEXT

SYNTAX:

SOFTEXT CNAME,PIX,VAR1,VAR2

ONE LINE OF ANY SOFTEXT EXCLUDING "<" AND ">

SOFTEXT AUTOMATICALLY CALLS THE CHARACTER SET CNAME INTO CORE IF NECESSARY AND THEN CREATES AN PIX CONSISTING OF ACTUAL 3D COORDINATES. VAR1 IS USED FOR SPACING BETWEEN CHARACTERS AND VAR2 IS USED FOR SIZE OF CHARACTERS. THE CHARACTERS ARE REREDRAW EVERY 1/10 OF A SECOND USING NEW VALUE OF VAR1 & VAR2. TO STOP RERDRAW OF PIX PUSH FS15.

ROM2 DOUBLE LINE ROMAN FONT (UPPER AND LOWER, COMPLETE PUNCTUATION)
SCR SINGLE LINE SCRIPT FONT (UPPER AND LOWER)
SCR2 DOUBLE LINE SCRIPT FONT (UPPER AND LOWER)
MAP SELECTED MAP SYMBOLS
ITALC LOWER CASE ITALIAN GOTHIC FONT
ITALU UPPER CASE ITALIAN GOTHIC FONT
ENGL LOWER CASE ENGLISH GOTHIC FONT
ENGU UPPER CASE ENGLISH GOTHIC FONT
PLAT LOWER CASE ROLLNO 50
EXAMPLES:  
SOFTEXT ITALL,TOMMY,DO,DI  
THIS IS AN EXAMPLE  
A=625  
SOFTEXT SCR2,SAM,A,DO  
G||GRAPHICS SYMBIOSIS SYSTEM  
A=300;B=1000;FSQN 15  
$A='HELLO';  
SOFTEXT ENGU,POGO,A,B  
+JELLO $A YELLOW  

THIS WILL DISPLAY "JELLO HELLO YELLOW".

NOTE THAT SPACES MUST BE ENTERED AS UPPER CASE CHARACTERS. PUNCTUATION IN THOSE SETS WHERE AVAILABLE IS INDICATED BY THE SAME PUNCTUATION CHARACTER BEING TYPED. MAP AND RUSU RUSL HAVE SPECIAL CORRESPONDANCES AVAILABLE FROM THE SYSTEMS PROGRAMMERS. DON'T USE '*' AS THE FIRST CHARACTER, SINCE THE SYSTEM WILL THINK IT'S A COMMENT.

//  **********************************************  
NO ONE TOLD ME ABOUT THAT COMMAND YET, SO HOW CAN I TELL YOU? BY THE WAY, ARE YOU SURE YOU SPELLED IT RIGHT?  
//  **********************************************  

SWITCHES:  
/X - TURNS OFF TRACING FOR THE VARIABLES SPECIFIED IN THE REST OF THE COMMAND.  

EXAMPLES:  
TRACE A,X,VE,FR  

TRACE/X FR,X  
THIS WILL TURN OFF THE TRACING OF FR & X.  

TRACE/X  
THIS WILL CANCEL TRACING OF ALL VARIABLES USED
NOTE: DIALS & ARRAYS CANNOT BE TRACED.
A MAXIMUM OF 20 VARIABLES CAN BE TRACED AT ONE TIME.

TEXT

SYNTAX: TEXT PIX,X,Y,Z
+ ONE LINE OF TEXT TO BE DISPLAYED
WHERE X,Y,Z ARE EXPRS

TEXT DISPLAYS VECTOR GENERAL HARDWARE CHARACTERS ON THE VG
SCREEN. THESE CHARACTERS (SPECIFIED BY PIX) ARE NOT STANDARD
PICTURES. THEY CANNOT BE ROTATED, PUTDSK'D, SCALED, ETC. AS
SOFTTEXT CHARACTERS CAN.

EACH PIX SIGNIFIES ONE LINE OF DISPLAYED TEXT.
X,Y,Z ARE EXPRS GIVING THE X,Y,Z COORDINATES OF WHERE PIX IS DISPLAYED ON
VG SCREEN. THEEXPR CAN RANGE FROM 2047 TO -2048. DEFAULT
PLACEMENT IS AT THE LEFT MARGIN OF THE VG SCREEN.

SWITCHES:
/1 THRU /4 - DISPLAYS PIX HORIZONTALLY ON VG SCREEN
WITH /1 BEING SMALLEST CHARACTER SIZE
AND /4 THE LARGEST. DEFAULT IF NOT
GIVEN IS /4.

/5 THRU /8 - DISPLAYS PIX VERTICALLY ON VG SCREEN
WITH /5 SMALLEST CHARACTER SIZE AND
/8 LARGEST.

SPECIAL SYNTAX:

IF A "I" IS IN DISPLAY LIST OF PIX THE NEXT
CHARACTER DISPLAYED IS IN UPPER CASE.
A "II" WILL LOCK DISPLAY IN UPPER CASE CHARACTERS
THIS IS NORMAL CHARACTER DISPLAY DEFAULT.
A " " WILL DISPLAY THE NEXT CHARACTER IN LOWER
CASE TYPE.
A " " WILL LOCK DISPLAY IN LOWER CASE CHARACTERS.

A WIDE RANGE OF SPECIAL CHARACTERS ARE AVAILABLE
WITH THIS COMMAND. SOME OF THESE INCLUDE MATHEMATICAL SYMBOLS
AND THE GREK ALPHABET. TO DISPLAY THESE SYMBOLS, THE OCTAL
CHARACTER CODE OF THE DESIRED SYMBOL IS ENCLOSED IN ANGLE
BRACKETS ("<","="). THE OCTAL CODE FOR ALL THE SPECIAL
CHARACTERS CAN BE FOUND IN THE VECTOR GENERAL GRAPHICS DISPLAY
REFERENCE MANUAL OR IN THE VECTORGRAPHICS II PROGRAM REFERENCE
CARD.

TO UNDERLINE CHARACTERS, FOR EXAMPLE, USE <10> FOR EACH CHARACTER:
I.E. HELLO<10><10><10><10><10> WILL UNDERLINE HELLO.

EXAMPLES:

TEXT SAM,2000,2000,1000
+HELLO
THIS WILL DISPLAY THE WORD 'HELLO'
1000 IN THE LARGEST CHARACTER SIZE.

TEXT/5 F A A,-2000,-1000
+<342> COMPUTER | GRAPHICS
THIS WILL DISPLAY VERTICALLY, STARTING
AT POSITION -2000,-1000,0. THE GREK LETTER
BETA (FROM THE OCTAL CODE <342>) FOLLOWED
BY THE STRING 'COMPUTER GRAPHICS' WITH
THE FIRST LETTER IN EACH WORD CAPITALIZED
WHILE THE REST OF THE STRING IS IN LOWER
CASE. THE ENTIRE LINE WILL BE IN THE
SMALLEST CHARACTER SIZE.

TEXT GA
+$A <275> <277>
THIS WILL DISPLAY IN UPPER CASE WHATEVER IS
IN STRING $A FOLLOWED BY THE SYMBOLS FOR
'NOT EQUAL' AND 'INFINITY'.
NOTE: $A CAN BE A MULTI-LINE STRING.
DO NOT USE A '*' AS THE FIRST
CHARACTER

// **** T R A C E D. ****

// **********************************************************

SYNTAX:

T I C K

T I C K IS THE SYSTEM TIMER. TICK 60 WILL WAIT ONE
SECOND (OR SIXTY FRAMES ON THE MOVIE CAMERA), TICK 240
WILL WAIT A SECONDS. ETC. TICKS IN DOLOPPEP MAPDES ARE LOCAL TO THE
*ZAP:<A=A+1
+SKIP -1>

IF IT IS VIP'D BECAUSE OF THE INFINITE LOOP,
THE MACRO IS NOT VIP-ABLE, DO NOT VIP THE MACRO

// *********************************************************************
/* CCG-HCC, WE DON'T HAVE IT
// *********************************************************************
// *********************************************************************
/* WAITFOR
SYNTAX: W AITFOR MNAM Eli
THE DOLOOPED MACRO CONTAINING THE "WAITFOR MNAMEI" WILL WAIT
FOR MNAMEI TO FINISH BEFORE IT CONTINUES. THIS COMMAND IS USEFUL FOR
SYNCHRONIZING DOLOOPED MACROS. IT CAN BE USED ON DOLOOPED MACROS
WITHIN DOLOOPED MACROS TOO;
SAM:< ****
DOLOOP FRED
WAITFOR FRED
UNLOOP FRED
SAM HAS TO BE DOLOOPED FOR THIS TO OCCUR. IF THE DOLOOP/WAITFOR
PAIR ON FRED ARE REPLACED BY A DO LOOP, THE WHOLE MACRO WILL
BE COUNTED AS A SINGLE LINE IN THE DOLOOPING STRUCTURE, WHICH MAY
OR MAY NOT BE MORE DESIRABLE THAN FRED BEING DOLOOPED AS WELL.
SEE HELP DOLOOP TOO.

// *********************************************************************
/* THAT ONE WAS A REAL WINNER, I NEVER HEARD OF IT... CHECK YOUR SPELLING
JUST TO MAKE SURE YOU DIDN'T MAKE A MISTAKE...!
// *********************************************************************
/* OK, YOU TELL ME WHAT WENT WRONG, TURKEY.... IF YOU DON'T KNOW ASK THE
NURD...!
// *********************************************************************
/* YA REAL WINN, I THOUGHT EVERYBODY WHO USES GRASS KNOWS THAT THERE
ISN'T ANY COMMAND THAT STARTS WITH A "Y", CHECK-A-YOU SPELLING!
// *********************************************************************
/* ZAPPOII

One tick unit equals 1/60 of a second.

Example:

- `PUTL PIX`
- `TICK 10`
- `GETL PIX`
- `TICK 10`
- `SK -4`

This will cause PIX to flash on and off 3 times a second.

// ******************************************************************
// ** TYPE 
// **
// ** TYPE DNAME.XXX XX.XX 
// **
// ** TYPE TYPES THE DNAME FROM THE AREA INDICATED BY "XX.XX" 
// ** IF NO "XX.XX" IS INDICATED, THE DEFAULT IS YOUR AREA.
// **
// ** EXAMPLES: 
// ** TYPE DRAW.MAC 
// ** TYPE PATHSDEC 
// **
// ** TYPING "Ic" AFTERS ABORTS THE DEFAULT IS YOUR AREA.
// **
// **
// ** EXAMPLES: 
// ** TREE 
// ** TREEXSYNTAX: TREE 
// ** TREE GIVES THE USER A DIAGRAM OF HIS DATA STRUCTURE. IT LISTS 
// ** THE PICTURES AND GROUPS IN THE DISPLAYED STRUCTURE BY NAME. 
// ** INDICATING GROUPING LEVELS AND HIERARCHIES BY TABS.
// **
// ** UNFILM 
// ** UNFILM ESOTERIC////////
// **
// ** UNFILM RESETS TO NORMAL RUN MODE AFTER FILMING WAS USED. ("U" IS 
// ** ENOUGH TO TRIGGER THIS COMMAND.)
// **
// ** UNLOOP 
// ** UNLOOP MNAME1,MNAME2,,..... ESOTERIC////////
// ** UNLOOP TAKES A MACRO OUT OF THE DOLOOP STRUCTURE. 
// ** SEE HELP DOLOOP FOR DETAILS.
// **
// ** VIP 
// ** VIP CPLNAME ESOTERIC////////
// ** VIP (VERY IMPORTANT PROGRAM) TAKES .CPL (COMPILED 
// ** MACROS) AND EXECUTES THEM AT A HIGH PRIORITY AT 60 TIMES 
// ** A SECOND OR ONCE PER TICK (IF YOU'RE FILMING, ETC.). 
// ** THE .CPL SHOULD NOT CONTAIN ANY NON-TERMINATING LOOPS UNLESS 
// ** YOU REALLY WANT THE SYSTEM TO CEASE NORMAL OPERATION QUICKLY 
// ** (INFINITE LOOPS AT HIGH PRIORITY LOCK OUT EVERYTHING ELSE). 
// ** VIP IS FOR CERTAIN ANIMATION USES AND TO ELIMINATE SOME 
// ** TYPES OF LOOPS AT *" LEVEL. IF AND IS DO NOT WORK ON 
// ** VIP'D MACROS.
// **
// ** EXAMPLES: 
// ** IF THE MACRO FILE LOOKS LIKE:
// ** #Sss:<a=do+di/256
// ** #b=d2*4>
// ** THIS WILL HAPPEN BY ITSELF EVERY 1/60 OF A SECOND 
// ** IF YOU:
// ** #COMPILE SSS.BIF
// ** #VIP BIF
WHEP

N,X,Y,Z,K

AR EXP

SNAP

DV

A POINT, GIVEN ONE TO

NEW

COORDINATES

SPECIFIED BY X,Y,Z. THE VALUE OF N CAN RANGE FROM 1, THE FIRST

POINT IN THE PICTURE, TO THE PICTURE'S LAST POINT.

K IS A VARIABLE IN WHICH THE FOLLOWING IS INDICATED:

K=0 DRAWN VECTOR

K=1 NON-DRAWN VECTOR (JUMP)

K=-1 END OF LIST (LAST VECTOR)

SWITCHES: /N - WORKS IN CONJUNCTION WITH

GETPOINT. ZAPPS THE LAST

POINT THAT WAS GETP'D.

BE CAREFUL NOT TO SET K TO -1 UNLESS YOU'RE SURE BECAUSE IT

TERMINATES THE LIST. ZAPPOI IS EFFECTIVELY THE REVERSE OF GETPOINT.

EXAMPLE:

ZAPPOI LAMP, 25, -1500, -1600, 500, 0

THIS WILL ZAP THE 25TH POINT OF LAMP TO THE

NEW COORDINATES SHOWN.

NOTE: ZAP CAN, IN FACT, BLANK THE LAST VECTOR (K=1).

HOWEVER, GETPOINT WILL STILL READ IT AS K=1 SINCE IT IS THE

END OF THE LIST. PUTDSK WILL ALSO GET CONFUSED IF THE LAST VECTOR

IS BLANCED SO DON'T PUTDSK A PICTURE WITHOUT RE-ZAPPING

THE LAST VECTOR TO K=-1.

AN EXAMPLE OF HOW TO USE GETP AND ZAP TOGETHER:

N=0

%POC N=N+1

GETP PIX, N, X, Y, Z, K

IF K=-1, ZAP PIX, N, X*2, Y*2, Z*2, K

SK %POC

THIS WILL DOUBLE THE SIZE OF THE PIX.

******************************************************************

ZAPTEXT

SYNTAX: ZAPTEXT PIX, $VAR1, $VAR2

ZAPTEXT SEARCHES THE PIX FOR $VAR1. DISPLAYED AS TEXT AND

REPLACES IT WITH $VAR2. IF IT IS NOT FOUND, THE COMMAND EXITS WITH AN

ERROR. THIS COMMAND DOES NOT ALWAYS WORK--COMPLAIN AND WE MAY FIX IT.

******************************************************************

NO DESCRIPTION IS AVAILABLE FOR THIS COMMAND AT THIS TIME.

******************************************************************

/*

NO DESCRIPTION IS AVAILABLE FOR THIS COMMAND AT THIS TIME.

******************************************************************

ERROR 1

1.E.PIX------------------------------------------> ERROR 1

THIS NAME CANNOT BE FOUND--CHECK SPELLING OR DIFCOR

ERROR 2

2.E.CMD----------------------------------------> ERROR 2

THIS COMMAND NAME CANNOT BE FOUND--CHECK SPELLING AND HELP

ERROR 3

3.E.SWH-----------------------------------------> ERROR 3

THIS SWITCH IS NOT DEFINED FOR THIS COMMAND

ERROR 4

4.E.SYN-----------------------------------------> ERROR 4

UNDIAGNOSTICABLE SYNTAX ERROR--SEE HELP COMMAND

ERROR 5

5.E.UNF-----------------------------------------> ERROR 5

MORE INFORMATION NEEDED FOR THIS COMMAND--SEE HELP COMMAND

ERROR 6

6.E.OVF-----------------------------------------> ERROR 6

TOO MUCH INFORMATION GIVEN--PLEASE LOG THIS ONE AS A RUG

ERROR 7

7.E.NID-----------------------------------------> ERROR 7

INPUT DEVICE NOT AVAILABLE

ERROR 8

8.O.NIF-----------------------------------------> ERROR 8

OUTPUT DEVICE IS NOT AVAILABLE

ERROR 9

9.E.NDF-----------------------------------------> ERROR 9

THIS NAME DOES NOT EXIST ON THE DISK--CHECK DIFDSK

ERROR 10

10.E.NOF----------------------------------------> ERROR 10

THIS NAME IS ALREADY ON DISK OR NOT LOGGED INTO 1,1

ERROR 11

11.E.CAR----------------------------------------> ERROR 11

COMMAND GOT CONFUSED WITH FUNNY CHARACTER--CHECK TYPING

ERROR 12

12.E.NCS----------------------------------------> ERROR 12

THIS FILE IS NOT A CHARACTER SET

ERROR 13

13.E.CCR----------------------------------------> ERROR 13

SYSTEM SPACED OUT--NO STORAGE LEFT--DELETE SOMETHING

ERROR 14

14.E.TYP----------------------------------------> ERROR 14

THIS NAME EXTENSION IS NOT VALID--SEE HELP GETDSK

ERROR 15

15.E.MAX----------------------------------------> ERROR 15

TOO MANY NAMES USED--DELETE SOMETHING

ERROR 16

16.E.DEV----------------------------------------> ERROR 16

A DEVICE OR VARIABLE NAME IS EXPECTED HERE

ERROR 17

17.E.NFX----------------------------------------> ERROR 17

THE PICTURE/MACRO DUALITY HAS BEEN CONFUSED--CHECK DIFDSK

ERROR 18

18.E.STR----------------------------------------> ERROR 18

THE DISK HAS NOT BEEN INITIALIZED FOR YOUR AREA--SEE TOM

ERROR 19

19.E.PRV----------------------------------------> ERROR 19

THIS PICTURE NAME ALREADY IN USE--CHECK DIFDSK OR RENAME

ERROR 20

20.E.TMT----------------------------------------> ERROR 20

THIS NAME IS NOT A TMT--DELETE SOMETHING

ERROR 21

21.E.STR----------------------------------------> ERROR 21

THE DISK NOT INITIALIZED--SEE TOM
MAXIMUM NUMRF (20) OF PUSHES EXCEEDED--POP SOMETHING
101.XXX------------------------ Error 65
ONLY EDITOR FILES MAY BE CLOSED'D
102.XXX------------------------ Error 66
DIRECTORY CANNOT BE DECODED ON THIS DEVICE
103.XXX------------------------ Error 67
FILE EXTENSION AND DATA TYPE DO NOT MATCH
104.XXX------------------------ Error 68
ONLY SWAP MODULES MAY BE MADE CORE-RESIDENT
105.XXX------------------------ Error 69
ILLEGAL VG CHARACTER SIZE 1 TO 8 ONLY ALLOWED
106.XXX------------------------ Error 70
POPPING WITHOUT PUSHING (ENOUGH) IS NOT MAKING SENSE
107.XXX------------------------ Error 71
A SWAP MUST BE SPECIFIED HERE
110.XXX------------------------ Error 72
FPP: ILLEGAL OPERATOR IN EXPRESSION
111.XXX------------------------ Error 73
FPP: INCOMPLETE EXPRESSION
112.XXX------------------------ Error 74
FPP: UNBALANCED PARENTHESIS
113.XXX------------------------ Error 75
FPP: ILLEGAL FLOATING CONSTANT
114.XXX------------------------ Error 76
FPP: INVALID OPERAND IN EXPRESSION
115.XXX------------------------ Error 77
FPP: NO ARGUMENT SPECIFIED ON FUNCTION CALL
116.XXX------------------------ Error 78
FPP: DIVISION BY ZERO ATTEMPTED
117.XXX------------------------ Error 79
FPP: FLOATING TO INTEGER CONVERSION ERROR
120.XXX------------------------ Error 80
FPP: ATTEMPT TO TAKE THE LN OF ZERO
121.XXX------------------------ Error 81
FPP: SQRT OF A NEGATIVE NUMBER IS IMAGINARY
122.XXX------------------------ Error 82
THIS NAME DOES NOT EXIST ON TAPE
123.XXX------------------------ Error 83
THIS NAME CANNOT NOW BE USED ON TAPE (MAY ALREADY EXIST)
124.XXX------------------------ Error 84
THE TAPE HAS NOT BEEN INITIALIZED FOR YOUR AREA - SEE TOM
125.F.CPL---------------------- Error 85
ONLY CPL FILES CAN BE 'EXECUTED'
126.XXX------------------------ Error 86
COMPILF: INVALID STATEMENT SYNTAX
127.XXX------------------------ Error 87
COMPILF: INVALID LINE END CP LINE SYNTAX, CHECK MACRO
130.XXX------------------------ Error 88
COMPILF G00FED, PLEASE LOG THIS RUG
131.XXX------------------------ Error 89
COMPILF WILL NOT PROCESS ALL OF YOUR MACRO - SEE TOM
132.XXX------------------------ Error 90
COMPILF: MISSING OPERATOR IN PUTPC1, GETPC1, OR ZAPC1
133.XXX------------------------ Error 91
COMPILF: ILLEGAL OPERATOR IN FLOATING EXPRESSION
134.XXX------------------------ Error 92
COMPILF: INCOMPLETE FLOATING EXPRESSION
135.XXX------------------------ Error 93
COMPILF: UNBALANCED PARENTHESIS IN FLOATING EXPRESSION
136.XXX------------------------ Error 94
COMPILF: ILLEGAL FLOATING CONSTANT
137.XXX------------------------ Error 95
COMPILF: INVALID OPERAND IN FLOATING EXPRESSION
138.XXX------------------------ Error 96
COMPILF: STRING LENGTH (8000 CHARACTERS) EXCEEDED
140.XXX------------------------ Error 97
ATTEMPT TO ACCESS NEGATIVE LOCATION IN STRING
142.XXX------------------------ Error 98
TARGET STRING IS NULL
143.XXX------------------------ Error 99
MATCH STRING IS NULL
144.XXX------------------------ Error 100
BUMP ONLY WORKS IF $A IS A-Z,VA-VZ,WA-WY,FA-FY,AA-AZ,LA-LY
145.XXX------------------------ Error 101
TARGET IS NOT A 'LINED' STRING
156.XXX------------------------ Error 110
ERROR IN POPPING--SEE HELP POPVAR
160.XXX------------------------ Error 112
COMPILF: NO ARGUMENT SPECIFIED ON FUNCTION CALL
161.XXX------------------------ Error 113
ARRAYS ARE AA-AZ ONLY
162.XXX------------------------ Error 114
UPPER BOUND OF THE ARRAY IS TOO LARGE OR SOMETHING
163.XXX------------------------ Error 115
YOU CANNOT EXECUTE STRINGS SHORTER THAN 8 CHARACTERS, SORRY.
164.XXX------------------------ Error 116
CC~NPILE: NESTED MACROS ARE NOT ALLOWED IN CPL FILES
T00 MANY SUBSCRIPTS HAVE BEEN SPECIFIED FOR THIS ARRAY
THE SYSTEM THINKS THIS IS AN ARRAY, BUT IT'S NOT
NOT ENOUGH SUBSCRIPTS ARE SPECIFIED FOR THIS ARRAY
ARRAY SUBSCRIPT OUT OF BOUNDS
THIS ARRAY HAS ALREADY BEEN DIMENSIONED
ARRAY SPECIFIED ISN'T BEEN DIMENSIONED--USE ARRAY COMMAND
THE VALUES OF DIALS AND ARRAYS CANNOT BE TRACED.
LOCAL VARIABLES CAN ONLY BE USED WITHIN MACROS.
THE MAXIMUM NUMBER OF TRACES HAS BEEN EXCEEDED.
YOU CAN'T DOLOOP THE SAME MACRO TWICE
MAXIMUM NUMBER OF THINGS TO DOLOOP IS 10
CANNOT UNLOOP SOMETHING THAT ISN'T DOLOOPEED
CANNOT WAITFOR SOMETHING THAT ISN'T DOLOOPEED
THIS ERROR IS IMPOSSIBLE
PUTDSK OF ARRAYS NEEDS SECOND ARGUMENT -- SEE HELP PUTDSK
CORE ALLOCATION IS CONFUSFED BY THIS DELETION--RESTART
SOMEHOW THE NAME TO BE DELETED WAS LOST--TRY RESTARTING
FIRST PICTURE IN GROUP COMMAND NOT DISPLAYED
SECOND PICTURE IN GROUP COMMAND NOT DISPLAYED
PIX TO BE GROUPED MUST BE ON SAME LEVEL (CONSULT TREE)
I FORGOT
I FORGOT
YOU CAN'T GET A LOCAL VARIABLE THAT DOESN'T EXIST YET
NOT ENOUGH SPACE FOR CCPY/V OR IS NOT COPIABLE
YOU CAN'T COPY COPIES--COPY THE ORIGINAL
PIX IS NOT PUTLINT' BUT ISN'T DISPLAYED EITHER--GIVE UP
CHECK THIS DISK FILE CAREFULLY--ERROR DETECTED IN WRITING
THE DISK FILE MAY BE INCOMPLETELY WRITTEN OUT
DELETE THINKS THIS PIX IS PUTLINT' BUT IT'S NOT--GIVE UP
GETPOINT IS CONFUSED BY PIX THAT AREN'T CLOSFD--CLOSF IT
GRASSR DOESN'T HAVE FLOATING POINT VARIABLES--USE GRASS/GRASS2
SOFTWARE POINTERS CRAPPED UP--RESTART (GOON)
SOFTWARE CRAPPED UP POINTERS--GIVE UP (UPCON)
SOFTWARE CRAPPED UP POINTERS--GIVE UP (UPRAD)
SOFTWARE CRAPPED UP POINTERS--GIVE UP (NEAR UPDOIT)
GROUPING SCREWED UP--GIVE UP (UPDCIT)
BAD CHAINING--GIVE UP (UPDOIT)
SYSTEM CAN'T HARDWARE DECHAIN--GIVE UP (HCHNCK)
PLEASE LOG THIS ERROR CAREFULLY IN THE BUG LOG--TCM