CATO

PRELIMINARY MANUAL

1. THE CATO DRIVER

To compile a CATO program:

1. Put CATO master on Unit 1
2. Autoload
3. Put program on desired input device
4. Type

```
CATO,I,L,O,P,S,M,E.
```

where
I = input medium
L = FORTRAN library medium
O = binary output medium
P = source language output medium
S = symbolic mode-switch output medium
M = MAP (symbolic machine language) output medium
E = error listing medium.

If argument O is a zero, the binary is written on tape 2 and an automatic "CATORUN,NAME,2,1,P." is executed.

To run a compiled program:

1. Select desired J-switch options for CATORES
2. Type

```
CATORUN,NAME,I,L,X.
```

where
NAME = the name given in the TITLE directive, or the name "PROGRAM" if no TITLE directive occurred.
I = binary input medium of CATO program
L = library tape from which to call CATORES
X = T if J-2 printout is desired on the typewriter
P if J-2 printout is desired on the line-printer

3. Follow directions given by typeouts. (Remember that if doping is selected, tape 2 is the dope tape.)

2. CHARACTERISTICS OF THE FORTRAN RESIDENT

I. When control rests at the typewriter, a low-pitched tone is heard. Instructions typed in from the typewriter will then be obeyed. When waiting to punch or to read paper tape, a high-pitched tone is heard. When a seventh level is read, the paper tape reader stops, and the high-pitched tone is heard. Depressing the reader character mode switch restarts the reader. When waiting to print on the line-printer, two alternating notes are heard.

II. Programs available without calling are:

1. CALL,N,A,B,.... Reads up to 10 binary programs into memory from medium n. Legal arguments are f,1,2,3, and 4. If f is the argument, reader must be in assembly mode.

2. CONTROL,I,O,C. Same as standard routine.

3. HOLD. Causes next call statement to preserve programs currently in memory.

4. INPUT,I,N. Same as standard routine.

5. OUTPUT,O,N. Same as standard routine.

6. BKSP,N. REWIND,N. LOCK,N. EFMARK,N. Basic tape handling subroutines. If $xN_1N_2..$ is used in place of N, the subroutine is executed for all media listed. REWIND,F. punches 6" of blank tape. REWIND,P. ejects a printer page.

7. PTMAR,I. Punches up to eight characters on paper tape. PTMAR. punches 2" of blank tape.

8. EDIT,I,O,C. Works like FORTRAN edit except that the edit tape is read completely into memory up to the bcd endfile (../..) before the edit is executed. Hence, EDIT,f,f,f. is possible. Columns 2-6 of the directive are interpreted in the same way as columns 1-5 of the COPYS directive.

9. COPY,I,O. VERIFY,A,B. TRANSFER,I,O,NAME. VFYLIB,A,B,NAME. These are identical to the standard routines.

10. SKIP,I,NAME. Skips tape until a record beginning with "..NAME" is read. SKIP,x. Skips to the first record beginning with "..".

11. COPYS,A,B,C. Copys medium A to B until the directive read from medium C is found. Columns 1-5 of the directive are compared to columns 1-5 of A. Spaces are ignored. When an identical record is found, the number in columns 7-9 of the directive is interpreted and that many more records are read

from A. All records but the last are copied onto B. Magnetic tape is backspaced over the last record read.

12. CATO and CATORUN. Described above, they are always available without calling.

13. RN. Has the same arguments as FORTBIN. If no arguments are given, they are assumed to be "f,1,0." This program executes an automatic

```
CALL,1,FORTBIN.
FORTBIN,I,L,O,P,M,E.
RUN.
```

14. CP,I,O. Executes an automatic

```
COPY,I,O.
(REWIND,I.  REWIND,O.)
VERIFY,I,O.
```

15. CLEAR. Clears all of memory above RESIDENT.

FURTHER CHARACTERISTICS OF RESIDENT

1. On paper tape input and output, a flex code "02", (question mark or underline) acts as a tab to column 73.

2. Autoloading with J-1 set automatically calls FORTBIN.

3. Autoloading with J-2 set automatically calls CATOCOM, so that it does not have to be called by the CATO driver.

4. Stepping, clearing the console and setting the program address to 10_8 returns the control to the console typewriter. Setting the program address to 20_8 has the same effect as autoloading.

3. LOGICOMP (THE LOGICAL SECTION OF THE COMPILER)

LEGAL DIRECTIVES

1. TITLE NAME, COMMENTS. Places the given name in the title word of the binary output. If no title statement occurs, the title is automatically PROGRAM.

2. MODE NAME. Specifies the start of the directives comprising a single mode.

3. CALC NAME1,NAME2,..NAME5(A,B,C,D),....,NAME10,.. Gives the names of the calculations to be executed when the keys listed in a preceding "KEYS" directive are pushed or the mode given immediately above is entered.

4. KEYS NAME1,NAME2,NAME3,... Gives names of keys to which the directives following the keys statement apply. The field of directives applying to these keys is terminated by another keys statement, a mode statement, or a program statement.

5. FORCE KEY NAME. Gives name of key to be forced after one of the keys specified in the previous keys statement is pushed.

6. NEXT MODE NAME. Gives the name of the mode which the student is to enter when one of the keys specified in the previous keys statement is pushed.

7. COPY MODE NAME. Specifies that the mode given in the preceding mode statement is to be identical with the mode whose name is given in the copy mode directive, except for the succeeding directives.

8. SAME AS NAME. Identical to the copy mode directive.

9. CHARACTER NAME1(X1,Y1)(X2,Y2)..., NAME2(XX1,YY1)(XX2,YY2).......
Gives the points (in octal) to be plotted for a given character. A character number is assigned to the specified symbol. To denote a space or character with no points, use ((,)). To denote the special carriage return character, use ((-)). Parentheses enclosing all the elements following a single name are optional, as are commas between individual elements (point pairs).

10. ASSIGN NAME1(KEY1,KEY2,..), NAME2(KEY10,KEY11,..),... Assigns a logical key number to the specified symbol and to its associated physical key(s).

11. GROUP GROUP1(NAME1,NAME2,..),GROUP2(NAME10,NAME11,..),... Assigns a logical number to a group of logical keys. A logical key name may appear in more than one group. A group name may appear in another group.

12. X. This directive (an X in column 7) is a do-nothing statement which is used following each END statement to improve the readability of the printout. An X directive must occur as the last statement in the program, following the FINIS statement.

FURTHER INFORMATION ABOUT THE DIRECTIVES

1. The internal numbers for mode names, character names, and assigned key names are the sequential numbers of the names in the order of occurrence, starting with zero.

2. All character and assign statements must precede the use of the key and character names in FORTRAN programs. Assigned key names and character names may be identical, but the logical number of the assigned key is used rather than the character number for identical names if the name is used in a calculation. The character number corresponding to a logical key number may be found by calling subroutine KEYCHAR.

3. If a key is not listed in a keys directive in a given mode, it is "ILLEGAL" and its use will cause the student keyset to be rung. (The PLATO subroutine RING is automatically entered as the calc for these keys.) The current mode will be re-entered.

4. A calc directive immediately following a mode directive will cause the specified calc to be executed each time the mode is entered, unless the mode was re-entered by an "ILLEGAL" key.

5. If a calc directive does not occur following a keys directive, a do-nothing calc, named NOCALC, is performed.

6. If a next mode directive does not occur following a keys directive, the next mode is automatically the current mode.

7. If a force directive does not occur following a keys statement, no key is forced.

8. One may not write calculations whose program statements give the names LISTARG, DELTARG, RING, or NOCALC. However, one may use the names RING and NOCALC in CALC directives. This is usually unnecessary, since, as described above, these routines are entered automatically in the appropriate situations.

9. PLATO FORMAT and STUDENT BANK statements may be placed among the logical statements rather than within a PROGRAM, if desired. These statements must occur before the first use in a calculation of the names they define.

4. CATOCOM (THE FORTRAN CALCULATION COMPILER)

Each CALC is written as a full FORTRAN program in normal format, complete with its subroutines (which may be used by any other CALC).

A CALC is begun with a PROGRAM, SUBROUTINE, or FUNCTION statement, which must contain a name. This is the name to which the CALC directives refer.

A program or a subroutine is terminated by a single END statement. Following the END may be either a new series of logical directives or another calculation or subroutine.

A FINIS statement or a second successive END statement terminates the entire CATO program.

LEGAL DIRECTIVES (See the FORTRAN manual for description of normal FORTRAN. All standard FORTRAN statements are legal and all standard FORTRAN rules apply.)

1. STUDENT BANK NAME1,NAME2,.. ,ARRAY1(N),ARRAY2(N),... This statement allows variables to be placed in a storage bank which is unique for each student. A name appearing in a student bank statement will cause reference to a storage location in a block of memory reserved for the student being processed. This block is duplicated for each student. Thus each student has his own private bank of variables.

   All student bank variables are initially zero.

   Indices of DO loops may appear in a STUDENT BANK statement. All variables which are specific or unique for each student, or which are referred to following a possible equipment-busy interruption must appear in a STUDENT BANK statement exactly once.

2. IN(KEY,GROUP)YES,NO. This statement is used to determine whether or not a logical key number is contained in a given symbolic key group. It must be followed by two statement numbers. Control will go to the first if the key is in the group, to the second if not.

3. PLATO FORMAT ARRAY, NPERWORD, A(SPACE)LIST(PERIOD). Packs character codes into an array. The array name may not appear in a DIMENSION or COMMON statement. The name is automatically placed in internal common. Once defined, the array may be referenced by any program or calculation. The total number of words generated by a PLATO format statement is

((TOTAL NO. OF CHAR-S)/NPERWORD + (1 IF REMAINDER, 0 IF NOT) + 1) .

The first word is a code word containing (NPERWORD - 1) in the upper address, (SHIFT PER CHARACTER) in the lower address.

NPERWORD is the number of characters to be packed per word. If the total number of characters in the program is

less-than-or-equal-to	2, NPERWORD must be	less-than-or-equal-to	48
"	4	"	24
"	16	"	12
"	64	"	8
"	128	"	6
"	256	"	6
"	512	"	5

NPERWORD is followed by a list of character names. If the names are only one letter long, they need not be separated from one another. If a name contains two or more characters, it must be set off from the others by parentheses.

SPECIAL VARIABLE NAMES USABLE IN CALCULATIONS

1. mode = NAME. Causes the next mode to be the mode whose name is specified. In effect, this overrides the next mode directive for one time.

2. force = NAME. Causes the forced key to be that whose name is specified. In effect, this overrides the force key directive.

3. ( ) = NAME. Where name is a logical key name, a character name, or a mode name, the internal numerical value of the symbol is used. If a logical key name and a character name are identical, the numerical value of the logical key is used. The character number corresponding to a logical key number can be found by calling subroutine KEYCHAR.

4. --- Logical key-group names may be used only within an IN statement.

5. xincrmt = N
   ( ) = XINCRMT*( ) + ( )

   Statements of the first type above cause the X increment used in plotting to be reset to the value N. It is wise to fix the X increment only once during the program, since it affects plotting for all students.

   Statements of the second type permit use of the value of the X increment for calculation of starting coordinates, etc.

6. yincrmt is the Y increment used for plotting. It may be used exactly like xincrmt.

7. xmaxmum is the X maximum for plotting. It may be used exactly like xincrmt.

8. ymaxmum is the Y maximum for plotting. It may be used exactly like xincrmt.

5. CATORES (THE RESIDENT PROGRAM FOR EXECUTION OF CATO PROGRAMS)

CATORES is supplied by CATO with a series of tables and constants which allow it to execute the logic and calculations written by the programmer. These are:

1. STORLIST. 32 words. Gives address of student bank beginning for each student.

2. CALCLIST. 128 words. Gives the addresses of CALCS 0 through 255, packed two to a word. Currently, only 254 calculations may be defined by a CATO program. This is an absolute maximum allowed number.

3. MODENTRY. N words. Number of the entry calculation for each mode.

4. KEYMODES. N words. This is the main logic switch of the CATO program. It is subdivided into equal blocks, one for each mode. Each mode is divided into sections, each section contains the calculation number, the force key number (if any), and the next mode number required by a specific logical key.

5. LENGTH. 1 word. Gives the number of words taken up by a single mode-block in keymodes.

6. STORAGE AND COMMON. 1 word. Gives the first and last addresses of student bank storage, common follows student bank.

7. CODTABLE. N words. Upper address equal character code corresponding to the n^{th} logical key number. Lower address equal logical key number corresponding to the n^{th} physical key number.

8. PLOTABLE. N words. Upper address equal the number of points in a given character. Lower address equal relative position in points table where the points for a given character are stored.

9. POINTS. N words. Character point information, packed four points to a machine word, each new character begins a new word.

10. GRPLIST. N word. A list of the addresses and lengths of the groups in GRPTBLS.

11. GRPTBLS. N words. A list of the logical key numbers comprising each group.

12. NUMEQUIP. 1 word. Number of student stations with which this program is to be run.

13. XINCRMT. 1 word. Gives the X base-coordinate increment to be used by PLOT, SELRASE, SPACE, and BKSP.

14. YINCRMT. 1 word. Gives the Y base-coordinate increment to be used by PLOT, SELRASE, SPACE, BKSP, NXTLINE, and PRVLINE.

15. XMAXMUM. 1 word. Gives the maximum desired X base-coordinate to be used by PLOT, SELRASE, SPACE, and BKSP.

16. YMAXMUM. 1 word. Gives the maximum desired Y base-coordinate to be used by PLOT, SELRASE, SPACE, BKSP, PRVLINE, and NXTLINE.

In execution, CATORES uses the following internal lists:

1. REQLIST. 128 words. This is the list of student key inputs (requests) that have not yet been fully processed. The first location is negative if there are no requests unfulfilled. If a request is present, the upper address equal logical key number corresponding to the physical key pushed. The lower address equal the student number. A partially executed request has the second bit from the left set. If the list becomes full (very unlikely) additional requests are ignored until a waiting request is completed.

2. JUSTDONE. 32 words. One word for each student.
   1st bit (left hand) set means there is a partially executed request for this student.
   2nd bit set means a base-point selection was just made.
   3rd bit set means the storage tube is in the write mode.
   4th bit set means a selective erase was just done.
   5th bit set means the screen was written upon since the last erase.
   The lower eight bits give the last CALC number for this student.

3. MODEBANK. 32 words, one for each student. Indicates the present mode of each student. MODEBANK is initially all zeroes.

4. LINKLIST. 513 words, 16 for each student plus a termination mark. Contains the links to each subroutine entered by the student. LISTARG stores links in LINKLIST (LISTARG is entered automatically) at the beginning of each calculation or subroutine. DELTARG removes the links from the list when the subroutine is completed. If more than 16 nested subroutines are used, student output will wait on this student until the sixteenth routine is completed.

5. VARILIST. 513 words, 16 for each student and a termination mark. Contains the arguments transmitted to each subroutine as it is entered. LISTARG stores the arguments at the same time it stores the links. DELTARG deletes the arguments from the list when it deletes the links from LINKLIST. If a total of more than 16 arguments are transmitted in nested subroutines, student output will wait on this student until the subroutine using the sixteenth argument is completed. The effect is almost unnoticeable. Student input is not affected, but output response may be delayed a fraction of a second.

When CATORES is entered, it checks first to see if the PLATO equipment is connected (a typeout occurs if it is not); then it interprets the jump switches and performs the initial procedures required by the jump switch options. If the doping option has been selected, a typeout occurs and the operator must type in the message which identifies the dope.

All of the student bank is then set to zero, the request list is set to -1 to indicate that no information is present, JUSTDONE and MODEBANK are set to zero, NOCALC is made the 0th CALC and RING the 255th CALC, ENDLIST and POINTER are initialized, and all table locations provided by CATO are substituted where required in CATORES.

Each student screen is erased and Slide 0 is selected. The clock is set to one one-sixtieth of a second and started. The clock is increased by one after each one-sixtieth of a second. Channel five, the PLATO input channel, is then activated and the computer waits for a student to press a key.

The initial mode is made zero. The entry calc (if there is one) is not performed until the mode is entered at some time after the CATORES initialization. Thus, if the first key hit in mode zero returns to mode zero, the entry calc will be performed when the next mode is set to zero, before the second key is pushed.

Note that if a key is illegal in a mode and returns to the same mode, the entry calc is not performed.

If a CATO systems output routine is used by a calculation, it may find that the student's equipment is not yet ready to accept a new output. In this case, the calculation is stopped, the place of stoppage and the arguments used are stored in tables, and CATORES looks to see if another student has a request waiting. Thus the same calc may possibly be used by another student before one student has completed it. When all request waiting for other students have been tried, the stopped request is restarted at exactly the point in the calculation where it left off. If the equipment is now found ready, the calculation is completed and the student enters his next mode.

JUMP SWITCH OPTIONS

1. No jump switches. Normal operation without doping. Program types DOPING SUPPRESSED. and proceeds automatically.

2. J-1. Set during entry and kept set during execution. Causes typewriter to type TYPE MONTH/DAY/YEAR/ID. The operator must then type the tag by which the run is to be identified. A sample message is 11/28/64/A. The slashes are required after each segment of the date. Only one identifying character may be typed after the third slash. The message may be terminated at any point by a period and, in case of typing error, may be restarted after a carriage return. Further examples of legal type-ins are

   2/22/44.   A.   TE/ST/.   JU/NE/8.   1/1/65/7.   ///Q.

   Student dope is collected at the time that each request reaches the computer (true record of time request occurred, untrue record of the mode of the student at the time the key was pushed, no force keys are recorded), and the student's dope is written on tape unit two, which must be ready to be written upon when the program begins.

   Unsetting J-1 during execution causes the dope tape to be terminated by an end of run record, an end of PLATO dope tape record, and a physical end-of-file. The computer stops ready to execute the restart program. The same results can be achieved (in case of accidental stopping of program) by starting manually at 4200B (see description of manual drivers).

3. J-2. Set until after the channel five active light comes on. Causes the program to rewind tape two before starting to write dope, if the doping option (J-1 or J-3 or both) has been chosen.

   If J-2 is not set before the channel five active light turns on, and either J-1 or J-3 is set, the program will skip to the end of tape two, which it now assumes to be a correctly terminated old dope tape, containing dope from previous runs.

   The tape may run away if it was not previously correctly terminated. This option allows the collection of dope from several runs upon a single tape.

FURTHER JUMP SWITCH OPTIONS

4. J-3. Set during entry and kept set during execution. Causes the same events as J-1 except that student dope is collected at the time of execution of the request, rather than at the time of input. (True record of the mode in which the key was obeyed, untrue record of the time at which request occurred, all force keys appear on the dope tape as real inputs.)

   Unsetting J-3 during execution causes the dope tape to be ended just as when J-1 is unset during execution.

   It should be noted that the dope written with J-3 set has a slightly different form from that written with J-1 set in that the key number has been increased by 400B. This does not affect the validity of the data becuase the largest possible key number is 177B.

5. J-1 and J-3 both. Set during entry and kept set during execution. Results in dope being recorded doubly and possibly randomly mixed. Each key is recorded twice, once when input and once when executed. Force keys are recorded only once. The data recorded during the execution phase and the force key data are distinguishable from the data recorded at the time of key input because the key numbers have been increased by 400B, exactly as they always are (see J-3 option description).

   Unsetting both switches together causes termination of the dope tape.

6. J-2. Set during execution. Causes an on-line printout to occur in the following form:

```
    STUDENT   KEY   MODE   CALC   *OCTAL

      000     023    005    021
      002     012    001    004
```

   and so on. Note that the numbers are given in "OCTAL" form.

   The printout will occur on the console typewriter if "CATORES" was entered with a zero in the accumulator. Otherwise (if the accumulator was non-zero) the printout occurs on the line-printer.

FORMAT OF THE DOPE TAPE AND OTHER PERTINENT INFORMATION

The dope for all students simultaneously is written upon tape two in 120 character BCD records.

If the physical end of the dope tape is reached, the tape is rewound with interlock, instructions are typed out, and the computer stops ready to go on. In this case, the tape is ended only with a physical end-of-file, to allow one to copy this tape to a longer tape, backspace over the end-of-file on the copy, and copy the remaining dope (assumed to have been put on a blank tape after the end-of-tape stop and subsequent continuation of the program occurred) on to the end of the longer tape.

In each 120 character record, there are seven 16-character blocks and a final 8-character block. The 16-character blocks are written in (02,03,03,08) as follows:

02 = octal student number
03 = octal key number (J-1 = true number, J-3 = number + 400B)
03 = octal mode number (J-1 = mode at time of input, J-3 = mode at time of execution)
08 = octal time in sixtieths of a second (J-1 = true time of input, J-3 = time of execution).

The final 8-character block is in (A8) format, apportioned as follows: A7 = the identification tag, up to seven characters in length, that was typed by the operator (see description of J-1 option).

11/28/64/A.	Results in	112864A	for these seven characters.
A.	Results in	0A	(followed by five BCD blanks)
1/1/64/7.	Results in	0101647	
TE/ST/S.	Results in	TESTOS	(followed by one BCD blank)
//A.	Results in	0000A	(followed by one BCD blank)
///Z.	Results in	000000Z	
G//UN.	Results in	0G00UN	(followed by one BCD blank)
.	Results in	00	(followed by five BCD blanks)

A1 = An indicator character which is

1. A BCD space (20B) for an ordinary dope record
2. A BCD slash (21B) for an end of session mark
3. A BCD T (23B) for an end of dope tape mark. This is always preceded by an end of session mark and followed by an end-of-file.

The clock is never zero except in a 16-character block which contains no information, so dope processers should ignore records with the time = 0. If a restart (using restart program) occurs, the clock is set back to one. Thus, a negative time change indicates that a restart has occurred.

MANUAL DRIVER OPTIONS

Use: Clear up and down

1. Set indicated number into program address register
2. Set accumulator as desired for options described below
3. Set door button, if necessary for the desired option
4. Hit START.

1. CATORES -- 04000. Resets all necessary tables and begins the execution of the program currently in memory.

   Accumulator options for CATORES:

   A. ACC = 0 J-2 printout occurs on console typewriter.
   B. ACC = (NON-ZERO) J-2 printout occurs on line-printer.

2. ENDOPE -- 04200. The program to end a dope tape on unit two. This causes all the necessary special records (end-of-session, end-of-tape, and end-of-file) to be written. This program stops ready to jump immediately to the restart program.

3. RESTART -- 04201. Program to restart a program which has been stopped for some reason. The dope tape must be on unit two and must have an end-of-dope-tape mark after the lesson which is to be restarted.

   Program scans through the dope tape three times to find the needed information. In case of equipment failure or an incorrectly ended dope tape, the typeout, "session not found.", may occur.

   When the program has succeeded in restoring all student screens to their terminal state, the typeout "Set desired J-switches. Hit start." occurs. One must set J-1 or J-3, whichever was set previously. Upon starting, the lesson or program may be continued from the point at which it was interrupted.

FURTHER MANUAL DRIVER OPTIONS

4. SPECTRE -- 04202. Simulation program executing CATO transcribed re-runs. Uses dope on tape unit two to simulate doped inputs. Tape two is never rewound. To use, type desired session tag and desired student numbers as indicated in initial typeouts. For example:

```
What session.
1/1/65/A.                                   (typed by operator)

What students.(in octal)
0/1/2/3/4/5/6/7/10/11/12/13/14/15/16/17/20.       "
```

In case of typing error, message may be restarted after a carriage return.

When the simulation of the indicated session is completed, program types "end of session." and stops ready to start at 04202B. If the session indicated is not on the dope tape, program types "session not found." and stops ready to start at 04202B.

Accumulator options for SPECTRE:

```
A. ACC = 0   Run at normal time rate.
B. ACC = 00000...00N   Run at the power of two in the lower
                       bits of ACC times the normal rate.
C. ACC = 40000...00N   Run at minus the power of two in the
                       lower bits of ACC times the normal rate.
```

```
Examples:  ACC = 00000000000000000000000000000002  means
                 run at four times normal speed.
           ACC = 40000000000000000000000000000001  means
                 run at one-half normal speed.
```

This program does not simulate simultaneous keyset inputs but is designed to simulate many students at once with only very slight changes in the original timing of outputs.

5. PROGSAVE -- 04203. Program to write memory onto tape four in the form of an autoloadable record. If J-2 is set, tape four is not rewound.

Accumulator options for PROGSAVE:

```
A. ACC = 0  when autoloaded, program stops ready to start
              at CATORES.
B. ACC = 1  ....starts at ACTIVATE (does not reset student
              tables).
C. ACC = 2  ....starts at SPECTRE.
```

6. PLATO SYSTEMS SUBROUTINES

These subroutines perform the indicated operation only for the student whose request is currently being executed.

1. CALL INTRUPT. Cause the calculation to be interrupted at this point to process requests from other students. Ideal for use with programs requiring time delays.

2. CALL KEY(I). Stores the logical number of the key being processed in "I".

3. CALL KEYCHAR(I,J). Takes the logical key number stored in I, finds the number of the character whose name is identical to the name of the logical key and stores that number in J. A zero is stored in J if there was no character whose name was identical.

4. CALL ERASE. Completely erases storage tube.

5. CALL RING. Turns on error light on student keyset. Light is turned off each time a key is pushed.

6. CALL EQUIP1. Activates special equipment 1.

7. CALL EQUIP2. Activates special equipment 2.

8. CALL EQUIP3. Activates special equipment 3.

9. CALL SLIDE(N). Selects slide number N.

10. CALL AUDIO(N). Selects audio effect number N

11. CALL SELRASE(X,Y,N). Selectively erases the storage tube at N sequential character positions, starting at (X,Y). When the series of selective erases has been completed, X, Y, and N are usually altered.

    N' = 0
    X' = X + N * XINC. Whenever X' > XMAX, X' is reset to 0. If M is the number of character positions remaining, X' = 0 + M * XINC.
    Y' = Y + (no. of times X' was reset to 0) * YINC. Whenever Y' > YMAX, Y' is reset to 0. If M is the number of additional times X' was reset to 0, Y' = 0 + M * YINC.

12. CALL PLOT(X,Y,CHARLIST,N,PACKLIST). Plots N sequential characters on the student's screen beginning at (X,Y). When the series of plots has been completed, N, X, Y, and CHARLIST(0) are altered.

N' = 0
X' = X + N * XINC. Whenever X' > XMAX, X' is reset to 0.
If M is the number of characters still to be plotted,
X' = 0 + M * XINC.
Y' = Y + (no. of times X' was reset to 0) * YINC. Whenever
Y' > YMAX, Y' is reset to 0. If M is the number of
additional times X' was reset to 0, Y' = 0 + M * YINC.
CHARLIST(0) = 0

X,Y, and N must be integer variables in the student bank. CHARLIST must be an integer FORTRAN array of at least one dimension and must be in the student bank.

If CHARLIST(0) is not changed before calling PLOT, it is automatically 0, since all of student bank is initially 0, and PLOT sets CHARLIST(0) to 0 each time a PLOT is completed. If CHARLIST(0) is 0 (note circumstances as noted above) when PLOT is called, PACKLIST is ignored and need not be included in the CALL statement. (CALL PLOT(X,Y,CHARLIST,N) is perfectly legal.) In this case, CHARLIST is assumed to contain right-justified character codes stored in reverse order to the direction of subscripting. Thus,

CHARLIST(1)	contains the last character to be plotted
CHARLIST(2)	contains the next-to-last character
- - - - - - - - - -	
CHARLIST(N-1)	contains the second character
CHARLIST(N)	contains the first character.

All locations of CHARLIST are unchanged when the plot is completed. If, however, CHARLIST(0) was set to -1 since the last PLOT for this student, the PACKLIST option will be selected. This causes N of the character codes packed in PACKLIST to be plotted. In this case CHARLIST(0) and CHARLIST(1) are used for temporary storage during plotting. Upon exit, CHARLIST(0) = 0 and CHARLIST (1) = character code of last character plotted. PACKLIST must be a FORTRAN array and must be included in the CALL statement. PACKLIST must have been packed by a routine (preferably a PLATO FORMAT or a CALL PACK(NPERWORD,PACKLIST, CHARLIST,N) statement) which places (NPERWORD-1) in the upper address of PACKLIST(0) and (SHIFT PER CHARACTER) in the lower address. Subsequent words of PACKLIST (from PACKLIST(1) on) must contain (NPERWORD) character codes in each word, packed in sequence from left to right, with the entire word left-justified if there are unused bits.

13. CALL PACK(NPERWORD,PACKLIST,CHARLIST,N). Packs N words of array CHARLIST (starting at CHARLIST(1)) into the array PACKLIST (starting at PACKLIST(1)). (NPERWORD) characters of CHARLIST are packed into each word of PACKLIST. (NPERWORD-1) is placed in the upper address of PACKLIST(0) and (SHIFT PER CHARACTER) is placed in the lower address.

14. CALL UNPACK(PACKLIST,CHARLIST,N). Unpacks N characters of PACKLIST into CHARLIST in reverse order, starting at CHARLIST(N). Thus, CHARLIST is suitable for use with PLOT. PACKLIST must have been packed by a routine which places (NPERWORD-1) in the upper address of PACKLIST(0) and (SHIFT PER CHARACTER) in the lower address.

15. CALL SPACE(X,Y,N). Spaces the coordinates N times, tests for XMAX and YMAX exceeded as PLOT does. X and Y are correctly changed upon exit. N is unchanged.

16. CALL BKSP(X,Y,N). Backspaces the coordinates N times, tests for X or Y zero, resets them back to XMAX or YMAX as required.

17. CALL NXTLINE(Y,N). Moves coordinate down N lines without changing X. Tests for YMAX exceeded, resets to 0 as required.

18. CALL PRVLINE(Y,N). Moves coordinates up N lines without changing X. Tests for Y = 0, resets to YMAX as required.