INTRODUCTION

PLATO, an acronym standing for Programmed Logic for Automatic Teaching Operations, is the name of a computer-controlled teaching system which is an extremely versatile and useable man-machine interaction system. The teaching system, containing highly flexible components controlled by a centrally located computer, automatically presents information to a student (or subject) and receives responses from him in an instructional sequence which utilizes the principal of feedback. The rate and manner of presentation of information to the student is determined by the author of the material presented and can be adjusted in accordance with the student's reactions and responses. Although the PLATO system was originally designed for teaching purposes, it quickly proved itself useful for research in many areas of education, behavioral science, physical science, medicine, etc.

The system includes four major parts:

1) The electronic hardware composed of a central computer, the student stations, and the communication links between student stations and the computer.

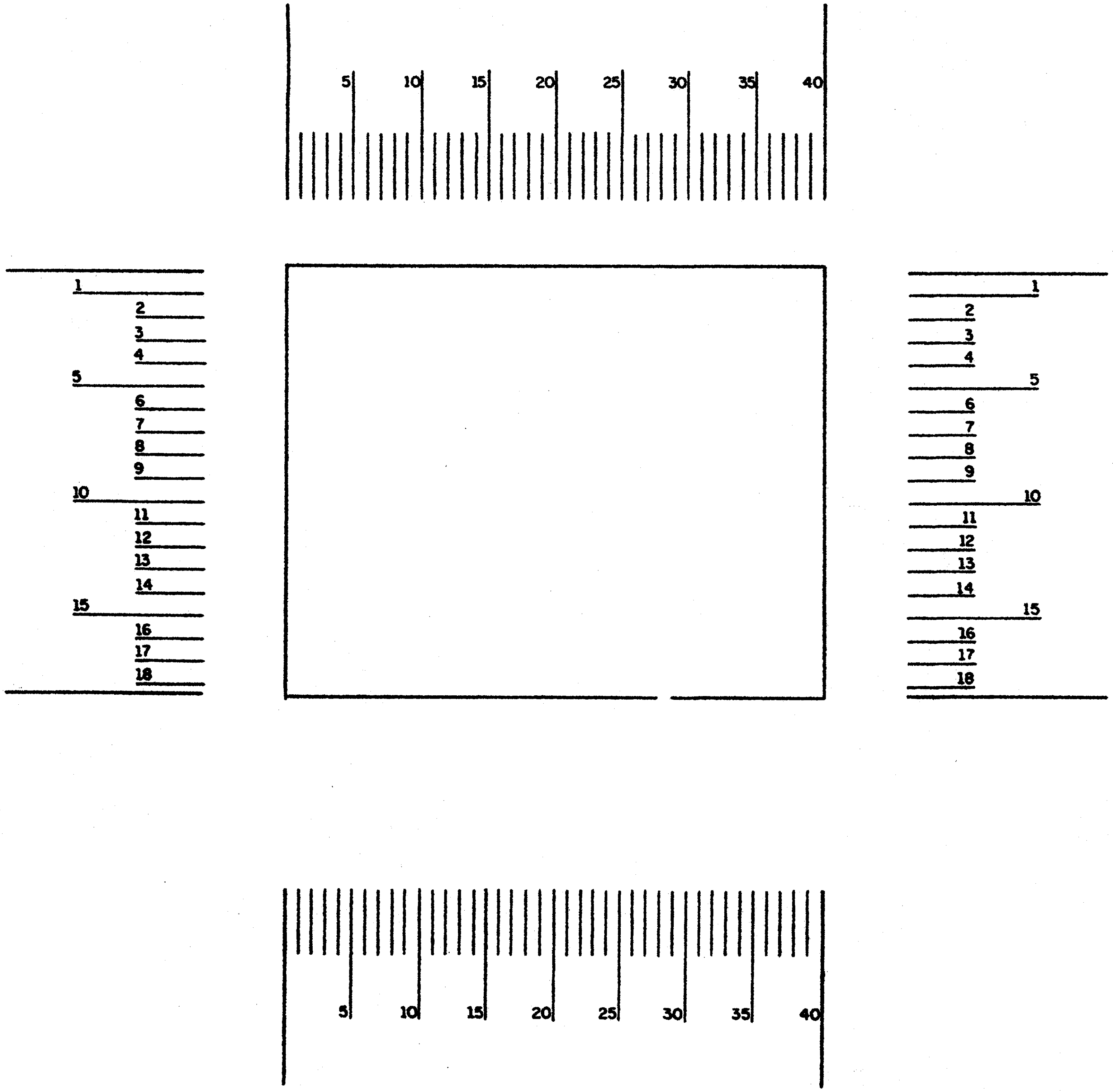
2) The computer program which determines the particular teaching rules (called "logic") to be used and records each student's progress, called "dope."

3) The lesson materials stored partly on film and partly in the computer, which utilize the various teaching logics.

4) The special effects used in the teaching system such as audio equipment, movie projector, physiological test apparatus, etc.

This operating manual for the PLATO system has been written as an aid to all users of the system from the non-technically-oriented to the experienced programmers. Part I contains a brief general description of the functions of the PLATO hardware (electronic equipment), its relationship to the PLATO software (computer program), the operation of the system by student input, the relationship of the teacher and/or programmer to the system, and remarks on the preparation of material for the system. Part II describes the general operating procedures for the PLATO system. Part III is a section designed for the PLATO teacher or programmer who is writing his own PLATO program using the PLATO Compiler (CATO). It is written in two parts, the first being detailed programming instructions, the second detailed compiling and editing instructions. Part IV discusses "doping" (student response records) procedures and analysis. The appendix, Part V, contains information which might be desired by the experienced PLATO programmer.

The PLATO Typing Form for Slides



For the PLATO camera, the material on each slide must fit within the rectangle illustrated above. Forms are available for typing copy, fitting illustrations and writing rough drafts.

The available typing area is 45 typed characters across (elite type) 18 lines. The same area plotted by the storage device is 40 characters across by 18 lines.

A. PLATO EQUIPMENT OPERATION

Turn On/Off Procedures

1. Equipment off:

   a. Check the engineers' card posted on the PLATO equipment. If the equipment may be turned on, push the "on" button. The time clock will be set automatically.

   b. After the equipment is turned on, push the storage device buttons for the student stations to be used.

   c. The first user after the equipment turn on must push the door button before autoloading to clear the equipment registers properly.

   d. When finished, turn the equipment off unless there will be another user in a short time.

2. Equipment already on:

   a. Check the time clock to see if adequate time is left for the run. (Black hand shows remaining time.) If not, push the "on" button to reset the clock.

   b. Check the storage device buttons for the student stations to be used.

   c. When finished, turn the equipment off unless there will be another user in a short time.

3. Unexpected equipment turn off:

   a. Check the time clock. (Black hand shows remaining time.) If time has run out, push the "on" button to reset the clock.

   b. Check for voltage drop. If the meter indicates a voltage drop, read the posted directions carefully. Push the reset button, if indicated.

   c. If neither a nor b is the case, summon a PLATO engineer.

Scanners

A. Using a program with slides:

1. Take the acetate sheet(s) from the storage cabinet. The slide sheets are named and numbered.

2. Open the scanner doors and lift the lens plates. There are catches on the inner sides of the scanner which must be moved aside before the plates can be raised.

3. Put the slide sheets,labeled side up, in their respective scanners, A or B. Be sure the holes on the sheets fit the pegs in the front of the scanner.

4. Lower the lens plates by pushing the catches aside. Check to see that the plates are all the way down. The scanners are turned on by a pressure button under the plates.

5. Close the scanner doors completely.

6. When the run is finished, remove the slide sheets and return them to the proper slot in the storage cabinet.

B. Using a program with no slides:

1. Open the scanner doors and lift the lens plates. There are catches on the inner sides of the scanner which must be pushed aside before the plates can be raised. When the plates are up the scanners are off.

2. Close the scanner doors completely.

B. COMPUTER PROCEDURES

Read-Ins

Before a run, check to see if the equipment is properly turned on, the right slides in, and the proper keycaps on the keysets to be used. Push the door button to clear the equipment registers if the equipment has been turned off for any length of time.

1. Starting a run using a binary program tape:

   a. Put PLATO Master tape (#100) on tape unit 1, DOPE tape (ring in) on unit 2, and the binary program tape (and data tape if used) on unit 3 or 4.

   b. Push the autoload button. Type: catorun, etc.; as described below.

      CATORUN,NAME,I,L,X. Puts the binary version of a PLATO program in memory ready to run.

      name = the name given in the title statement, or the name "program" if no title statement exists
      i = input medium for the binary version of the program
      l = input medium for the PLATO Master tape
      x = t if J-2 printout (DOPE) is desired on the typewriter
      p if J-2 printout (DOPE) is desired on the line-printer

   c. When the console lights up, set J-1 or J-3 for the desired DOPE option, and J-2 if using a fresh DOPE tape.

   d. Hit Start. The typewriter will ask for DOPE identification tag, MONTH/DAY/YEAR/ID. Type the identification tag.

   e. If J-2 was set,unset it either after typeout MONTH/DAY/YEAR/ID has occurred or after typing a message in.

   f. Check to see if the channel 5 light is on. If so, the program is ready to run. Put the production sign, PLATO, on the console desk.

2. Starting a run using an autoloadable program tape:

   a. Put the autoloadable program tape on tape unit 1, DOPE tape (ring in) on unit 2, and the data tape (if used) on unit 3 or 4.

   b. Push the autoload button. Check to see if the "function code" and the "execution address" read "76 0 04000." If so, follow the instructions in c through f above.

3. Continuing a run using a PROGSAVE'd program tape:

   a. Put the autoloadable program tape on unit 1, DOPE tape (ring in) on unit 2, and the data tape (if used) on unit 3 or 4.

   b. Push the autoload button. Check to see if the "execution address" reads "4201" (ready to transfer to RESTART). If the "execution address" is <u>not</u> 4201 (this will be the case if the "accumulator" was <u>not</u> set to 1 at the end of the previous run) simply reset 4201 in the execution address.

   c. Set J-1 or J-3 for the desired DOPE option, and J-2 if using a fresh DOPE tape. The DOPE tape will be positioned automatically.

   d. Hit Start. The typewriter will ask for DOPE identification tag, MONTH/DAY/YEAR/ID. Type the new identification tag.

   e. If J-2 was set, unset it either after typeout MONTH/DAY/YEAR/ID has occurred or after typing message in.

   f. Check to see if the channel 5 light is on. If so, the program is ready to run. Put the production sign, PLATO, on the console desk.

4. Ending a run (PROGSAVE'ing)

   a. Unset J-1 or J-3 to end the DOPE tape properly.

   b. Clear up and down.

   (To make an autoloadable program tape for the continued run, follow c through f below.)

   c. Mount the tape on which the saved program is to be written on unit 4, with ring in.

   d. Set the "accumulator" to 1 and the "program address" to 4203. Start. (Unless J-2 is set, tape 4 will be rewound before writing.)

   e. Unload tapes after tape 4 is written, take the ring out, and put the tapes away.

   f. Turn the equipment off.

Doping

A. Use of DPLIST1:

1. Put the PLATO Master tape on tape unit 1, the DOPE tape on unit 2 and the flextape of DPLIST1 in the reader in assembly mode. DPLIST1 is a binary flextape which causes the printout of key inputs by name vertically on the printout page.

2. Autoload.

3. Type: call,f.

4. Put the parameter tape in the reader in character mode.
(The parameter tape is a data tape of the key names in the order of their appearance in the assign statement(s) of the source program. This data tape is typed one name per line, left-justified and terminated by a BCD end-of-file.)

5. Put J-1 up if the entire tape is to be processed.
Leave J-1 down if only one session (1000 keys) is to be processed.

6. Type: run.
   or
   dplist1.

7. If J-1 was left down, type a carriage return and the session identification tag leaving out the slashes. After typing the tag, type another carriage return, and the processing will start.

Example:

```
cr
0724651   (For an original tag of 7/24/65/1.)
cr
```

If more than one session or more than 1000 key inputs are desired, put J-1 up anytime during the printer output for an addition 1000 key inputs. This procedure may be repeated for as many additional sets of 1000 key inputs as needed.

Emergency Procedures

A. Forgotten ENDOPE

1. When the DOPE tape position has been disturbed, but not the memory (e.g., a "reverse" or "rewind" button on the tape unit has been pushed without unsetting the jump switch):

   Position the dope tape by typing: copy,2,x.

   After the error printout, manually set the "program address" to 4200 (the entry to ENDOPE). This causes all the necessary termination marks to be written on the DOPE tape. The ENDOPE program stops ready to jump to the RESTART program.

2. When both the DOPE tape position and the memory have been disturbed:

   Position the DOPE tape by typing: copy,2,x.

   After the error typeout, type: efmark,2. (This will write an end-of-file on tape unit 2.)

   Note, however, that the end-of-session mark will not be recorded and the last record on the DOPE tape will be destroyed upon restarting a continued run.

B. Forgotten PROGSAVE

1. Put the autoloadable tape which was used for the previous run on tape unit 1, the Dope tape on unit 2.
2. Autoload.
3. Set the program address equal to 4202 (the entry to SPECTRE). Enter the proper values in A and Q (See Part II on reviewing using SPECTRE). This is a fast reading of the previous DOPE.
4. Start.
5. After a typeout END OF DOPE has occurred, proceed to PROGSAVE on tape unit 4 by setting the "program address" equal to 4203. Set ACC = 1, then Start. (See Part II-B, Ending a run.

C. Voltage drop

A voltage drop in the PLATO equipment may cause loss of information. See Part II-A, Unexpected equipment turn off.

Reviewing a Student Run Using SPECTRE

1. Put the PLATO Master tape on tape unit 1, and the written DOPE tape on unit 2.

2. Autoload.

3. Set the "program address" to 4202 (the entry to SPECTRE).

4. Set the "accumulator" for the desired option:

ACC = 0	Run at the normal rate.
ACC = 000...0N	Run at the power of 2 in the lower bits of the accumulator times the normal rate.
ACC = 400...0N	Run at minus the power of 2 in the lower bits of the accumulator times the normal rate.

   Example

   ACC = 0000000000000002 means to run at four times the normal speed.
   ACC = 4000000000000001 means to run at 1/2 the normal speed.

5. Set the "Q-register" for the desired option:

Q = 0	Clear both student bank and common.
Q = 1	Save common, but clear student bank.
Q = 2	Save student bank, but clear common.
Q = 3	Save both student bank and common.

6. Hit Start. The typewriter will type WHAT SESSION.

7. Type the dope identification tag just as it was originally typed for the actual run.

   Example Type: 1/3/65/a. (For the original tag 1/3/65/a.)

8. The typewriter will then ask WHAT STUDENTS.(IN OCTAL). Type the desired student number in octal.

   Example Type: 0/1/2/3/4/5/6/7/10/11. (For students 1,2,3,4,5,6,7,8,9.)

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

SPECTRE does not stop at the end of the specified session, but continues until it hits the end-of-dope-tape mark, at which point the typeout "end of dope" will occur. To stop before the end, simply use the step switch. If the session indicated is not on the dope tape, the program types "session not found" and stops ready to start at 4202b (the entry to SPECTRE).

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 to the PLATO equipment.

A. PROGRAM CONSTRUCTION

PLATO-Modified FORTRAN

CATO is a modified FORTRAN '60 compiler with additional features to accommodate the need in teaching operations. These additions are listed in this section of the manual under PLATO Statements, with examples.

In addition to the usual FORTRAN subroutines, there are many PLATO system routines which are currently available. More subroutines will be added to the library in the future as the need arises.

With the knowledge of FORTRAN programming, it is a relatively easy matter to write a complete computer program for use with the PLATO system. The following section assumes knowledge of FORTRAN.

Naming of Variables

All names created in the program statements must be standard FORTRAN names, beginning with a letter and consisting of from one to seven alphanumeric characters. All character, key, mode and calc names are treated as integers regardless of the first letter. All other names must follow the Fortran rules for fixed and floating point operations.

The following names may not be used as calc names in PLATO programs:

AUDIO	NXTLINE	SPACE	MODE
BKSP	PACK	UNPACK	KEYS
ERASE	PLOT	WRITE	CALC
EQUIP1	PLOTG	NOCALC	NEXT
EQUIP2	PRVLINE	LISTARG	SAME
EQUIP3	READ	DELTARG	FORCE
INTRUPT	RING	DRV00	COPY
KEY	SELRASE	:	all Master Tape routine names
KEYCHAR	SLIDE	DRV77	

It is not advisable to use identical names for mode and calc. The PLATO Compiler will accept identical names; however, one can run into trouble when the calcs are called.

Do not use single letter variable names in calculations, since these are generally reserved for the names of alphabet characters and keys.

Program Organization

```
      Title Statement
      Character List
           :
           :
      Assign List
           :
      Group List
           :
      Modeswitch Statements
           :
           :
      PLATO Format Statements     (These statements may also be placed
           :                       inside calculations.)
           :
      Student Bank Statement(s)   (This statement may also be placed
           :                       inside calculations.)
      Program Name1          \
           :                  |
      end                     |
      x                       |
      Program Name2           |
           :                  |
      end                     |
      x                       |
      Function F1              >  PLATO Calculations
           :                  |
      end                     |
      x                       |
      Subroutine S1           |
           :                  |
      end                     |
      x                       |
      Program Last            |
           :                  |
      end                     |
      finis                  /
      x
..
..
```

Sample Program

This is a very simple PLATO program showing the organization in actual statement form. Detailed information about the statements follows in this section of the manual.

```
999   title explain
c     this lesson shows slide 3 and plots plato when key go is pushed
898    character dmy((,)),p((30,44)(31,44)(32,44)(33,44)(34,44)(35,44)
     1(36,45)(37,46)(37,47)(36,50)(35,51)(34,51)(33,51)(32,51)(31,51)
     2(30,45)(30,46)(30,47)(30,50)(30,51)(30,52)(30,53)(30,54)(30,55)
     3(30,56)(30,57))
897    character l((30,44)(30,45)(30,46)(30,47)(30,50)(30,51)(30,52)
     1(30,53)(30,54)(30,55)(30,56)(30,57)(31,57)(32,57)(33,57)(34,57)
     2(35,57)(36,57)(37,57))
896    character a((30,57)(30,56)(30,55)(30,54)(30,53)(30,52)(30,51)
     1(30,50)(30,47)(30,46)(31,45)(32,44)(33,44)(34,44)(35,44)(36,45)
     2(37,46)(37,47)(37,50)(37,51)(37,52)(37,53)(37,54)(37,55)(37,56)
     3(37,57)(31,51)(32,51)(33,51)(34,51)(35,51)(36,51))
895    character t((30,44)(31,44)(32,44)(33,44)(34,44)(35,44)(36,44)
     1(37,44)(40,44)(34,45)(34,46)(34,47)(34,50)(34,51)(34,52)(34,53)
     2(34,54)(34,55)(34,56)(34,57))
894    character o((30,54)(30,53)(30,52)(30,51)(30,50)(30,47)(31,45)
     1(33,44)(34,44)(35,44)(37,45)(40,47)(40,50)(40,51)(40,52)(40,53)
     2(40,54)(37,56)(35,57)(34,57)(33,57)(31,56))
710    assign dmy(140),go(101)
603    plato format sentl,8,plato
524    student bank ix,iy,karlist(5),nchar
481    mode intro
       keys go
       calc show
390    program show
       call erase
       call slide(3)
       ix = 96
       iy = 70
       nchar = 5
       karlist(0) = -1
       call plot(ix,iy,karlist,nchar,sentl)
       end
       finis
       x
..
..
```

B. PLATO STATEMENTS

The Title Statement

Form:

```
     TITLE NAME
```

Example:

```
999  title arch
c    this lesson demonstrates archimedes principle
c    using inquiry logic
          :
          :
```

The title statement places the given name of the lesson in the title word of the binary version of the program. If no title statement occurs, the title is automatically PROGRAM.

Each PLATO lesson should have a title statement to facilitate calling the lesson by name from magnetic tape. In binary form, several lessons can be "stacked" on a single tape and are called individually using each unique name.

Character-Symbol Definition

The Character Statement

Form:

```
      CHARACTER NAME((X,Y)(X,Y)(X,Y)(X,Y)(X,Y)(X,Y)(X,Y)
     1(X,Y)(X,Y)(X,Y)(X,Y)(X,Y).......(X,Y))
```

Example:

```
805   character a((30,57)(30,56)(30,55)(30,54)(30,53)(30,52)(30,51)
     1(30,50)(30,47)(30,46)(31,45)(32,44)(33,44)(34,44)(35,44)(36,45)
     2(37,46)(37,47)(37,50)(37,51)(37,52)(37,53)(37,54)(37,55)(37,56)
     3(37,57)(31,51)(32,51)(33,51)(34,51)(35,51)(36,51))
```

Each character statement gives the points, in octal, which form a character to be plotted in a lesson. The X and Y coordinates for each point are listed and enclosed by parentheses. Another set of parentheses enclose the entire set of points for each character.

The length of each character statement is limited to 9 continuation statements as in regular FORTRAN, but there is no practical limit to the number of characters in the list. Several characters may be listed in the same character statement if the number of continuations does not exceed nine.

Example:

```
823   character o((30,54)(30,53)(30,52)(30,51)(30,50)(30,47)(31,45)
     1(33,44)(34,44)(35,44)(37,45)(40,47)(40,50)(40,51)(40,52)(40,53)
     2(40,54)(37,56)(35,57)(34,57)(33,57)(31,56)),p((30,44)(31,44)
     3(32,44)(33,44)(34,44)(35,44)(36,45)(37,46)(37,47)(36,50)(35,51)
     4(34,51)(33,51)(32,51)(31,51)(30,45)(30,46)(30,47)(30,50)(30,51)
     5(30,52)(30,53)(30,54)(30,55)(30,56)(30,57))
```

To denote a space or a character with no points, use ((,)). To denote the special carriage return character, use ((-)).

A character number is assigned to each character according to its position in the list. The first character (position 0) should be a dummy.

Example:

```
899   character dmy((,)),a((30,57)(30,56)(30,55).....
```

For more detailed information about character plotting, see Appendix B.

The Assign Statement

Form:

```
     ASSIGN NAME1(octal input number),NAME2(octal input number),...
```

Example:

```
734   assign go(056),stop(073),a(101),b(102),m(116),kerase(177),n1(061),
     1n2(062),n3(063),n4(064),n5(065),n6(066),n7(067),n8(070),n9(071),
     2n0(060),kspace(040),comma(074)
```

Each key on the PLATO keyset sends an octal input to the computer when the key is pushed. The assign statement gives a symbolic name to each of these inputs used in a lesson. The keyset has 96 unique inputs to the computer and the maximum number of assigns is 128. Octal input 000 should be avoided since it is the input of every key when the break button is depressed.*

A logical key number is given to each key according to its position in the assign list. If PLATO routine TRANSFR is used in the lesson, the first assign must be a dummy.

Example:

```
                                               (dmy is logical key number 0.
732   assign dmy(140),go(056),stop(073),...      go  "     "     "    "   1.
                                               stop  "     "     "    "   2.)
```

A character and a key may have the same name, but in calculations the distinction must be made clear by the use of PLATO routine KEYCHAR, or the key number will take precedence over the character number.

Each octal input may have only one name, but a name may have more than one octal input.

Example:

```
740   assign go(056,177)          correct

740   assign go(056),go(177)      incorrect- go(177) will replace go(056)

740   assign go(056),cont(056)    incorrect
```

For further information about the PLATO keyset, see Appendix C.

---

* The break key disengages all the keys and is commonly used to prevent disruption of the program while keycaps are being attached to the keyset.

The Group Statement

Form:

```
 GROUP GROUP1(NAME1,NAME2,NAME3,NAME4,NAME8),GROUP2(NAME10,
1NAME15,GROUP1,NAME20)
```

Example:

```
644  group alpha(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z),
    1num(n0,n1,n2,n3,n4,n5,n6,n7,n8,n9),alphnum(alpha,num),alfpunc(cr,
    2period,comma,qmark,hyphen,alpha)
```

The group statement partitions the keyset in any number of ways in addition to the listing of individual keys. Grouping is convenient when the same calculation is performed for several different keys and when a program decision depends on a key's membership in a particular group.

A key name may appear in more than one group, and a group name may appear in another group. The nesting of groups inside other groups has no practical limit as long as all the elements of the groups are clearly defined.

Modeswitch

The modeswitch section of a source program describes the teaching rules for a lesson. The "logic", as the teaching rules are called, is a series of modes in which various keys available to the student cause computer and system operations to be performed. Each mode is a listing of the keys and their associated calculations which are "legal" (available) at a particular point in the lesson.

Example:

```
100   mode intro
      keys cont
      calc nxtpage
      next mode one
```

The mode described above is named "intro." It has one legal key, "cont", which when pushed during a student run will trigger its associated calculation, "nxtpage." The calculation, a FORTRAN program, located elsewhere in the source program, increases the slide number by 1 and displays the next slide. When the calculation, "nxtpage", is completed the student is in the next mode, "one", with a new set of available keys and calculations.

All modes, except the first one which is mode 0, may occur anywhere in the modeswitch. The first mode is the starting point of the logic and is the first one entered at the beginning of a student run.

Logical mode numbers are assigned to the modes in the order of their appearance in the modeswitch.

The Mode Statement

Form:

```
     MODE NAME
```

Example:

```
100  mode intro
         .
         .
         .
         .
         .
101  mode one
         .
         .
         .
         .
102  mode two
         .
         .
         .
         .
         .
```

Each mode statement specifies the start of the statements which comprise a single mode. Each mode is terminated by the start of the next, and the last mode is terminated by any legal statements other than modeswitch statements.

The Keys Statement

Form:

```
      KEYS NAME
   or
      KEYS NAME1,NAME2,...
```

Example:

```
100   mode intro
      keys cont       (a single key)
           :
      keys alpha      (a group of keys)
           :
      keys n1,n2,n3   (several keys)
```

Each keys statement in a mode gives the name of a key which is "legal" (available) in that mode. Only those keys which are legal need to be listed. If a key is not listed and therefore "illegal", the PLATO routine, RING, which causes the keyset light to flash, is automatically the calculation for that key and the current mode is re-entered.

The field of statements applying to a key is terminated by another keys statement or by a mode statement.

The Calc Statement

Form:

```
        CALC NAME
    or
        CALC NAME1,NAME2,...
```

Example:

```
102     mode two
        keys cont
        calc nxtpage
             .
             .
             .
        keys go
        calc numpage, plotnum
```

Each calc statement gives the name of the calculation, a FORTRAN program, to be executed when the key listed in the preceding keys statement is pushed. If a calc statement is not listed after a keys statement, the PLATO routine, NOCALC, is automatically the calculation for that key.

Logical calc numbers are assigned to the calcs in the order of their appearance in the modeswitch. Any calc in the lesson may be called by another calc.

Two successive calc statements associated with a keys statement will result in the overwriting of the first calc by the second.

The Entry Calc

Example:

```
102     mode two
        calc setup
             .
             .
             .
```

A calc statement immediately following a mode statement will cause the specified calculation to be executed each time the mode is entered, unless the mode is re-entered by pushing an "illegal" key.

An entry calc in the first mode of the modeswitch will not be executed on initial entry to the mode at the start of a student run. It will be executed, however, on re-entry when a "legal" key is pushed which does not change modes or when the first mode is entered from another mode.

The Next Mode Statement

Form:

```
NEXT MODE NAME
```

Example:

```
100    mode intro
       keys cont
       calc nxtpage
       next mode one
            .
            .
            .
101    mode one
```

Each next mode statement gives the name of the mode to be entered when the key in the preceding keys statement is pushed and the calculation (if any) associated with that key is completed.

If a keys statement does not have a next mode statement associated with it, the current mode is re-entered automatically.

Mode changes can also be made inside FORTRAN calculations. See the section, Executable Logical Statements.

The Force Key Statement

Form:

```
     FORCE KEY NAME
```

Example:

```
104  mode m40
     keys n7
     calc metal
     force key n5
     next mode m43
          .
          .
          .
107  mode m43
     keys n5
     calc medium
     next mode m50
```

The force key statement gives the name of the key to be forced after the key specified in the preceding keys statement is pushed. In effect, the force key request is put into the request list ready for processing just as if the student had pushed the key.

In the above example, pressing key n7 while in mode m40 will execute the calc, "metal", and cause key request n5 to be put in the request list. The next mode, "m43", is then entered and key n5, already in the request list, immediately triggers the calc, "medium."

Keys may also be forced inside FORTRAN calculations. See Executable Logical Statements in this section of the manual.

The Copy Mode Statement

Form:

```
      COPY MODE NAME
```

Example:

```
103   mode three
      copy mode two
      keys alpha
      calc alfplot
      next mode four
```

The copy mode statement creates another mode like the mode whose name is given in the copy mode statement. The statements following the copy mode statement are additions and/or exceptions to the mode being copied.

Example:

```
102   mode two                  Mode two has 3 keys and 2 mode changes
      keys cont
      calc nxtpage
      next mode four
      keys alpha
      calc alfplot
      keys num
      calc numplot
      next mode four
103   mode three                Mode three has the same 3 keys, but 3 mode
      copy mode two                                                changes
      keys alpha
      calc alfplot
      next mode four
```

The copy mode statement can be used to add keys and calcs, replace calcs and to add and replace next modes.

The Same As Statement

Form:

```
      SAME AS NAME
```

The same as statement is identical to the copy mode statement.

Student Bank

The Student Bank Statement

Form:

```
STUDENT BANK NAME1,NAME2,.....ARRAY1(N1),ARRAY2(N1,N2),NAME4,...
```

Example:

```
200  student bank ix,iy,nchar,karlist(16),kount1,jflag,jpp,nkey,
    lns,itim,jarray(4,5),kount2
```

The student bank statement provides a block of memory storage for student variables. This storage bank is duplicated automatically to give each student his own unique bank of variables.

All variables which are specific or unique to a student and variables which are referred to following a possible equipment-busy interruption of an output calculation must appear in a student bank statement. Since a calculation may be interrupted during output of information to the PLATO equipment, the variables for the interrupted student must be saved in his bank to avoid changes made for another student using the same calculation.

Indices of DO loops may appear in the student bank and must appear there if an output of information is made inside the loop. Student bank arrays may have as many as three dimensions.

Student bank location(0) is automatically the number of the next mode, and location(1) is automatically the force key, if any. If there is no force key, location(1) equals 255. The names defined in the student bank statement begin at location(2).

Student bank statements may be placed before the FORTRAN calculations or inside an individual calculation before the first use of the names defined. The value of every student bank variable is initially 0.

The beginning address of the student bank for the student whose key is currently being processed is contained in index register 6.

PLATO Calculations (FORTRAN Programs)

Each calculation is written as a full FORTRAN program, subroutine, or function* in normal format, complete with its subroutines which may be used by any other calculation. A calculation is begun with a Program, Subroutine, or Function statement which must contain a name. This is the name to which the Calc statement in the modeswitch refers. Names of PLATO system routines may not be used as calculation names (See Part III on Program Construction-Naming of Variables).

A program, subroutine, and function must be terminated by a single End statement and an x. The x statement is a do-nothing statement which follows each End statement to improve the readability of the print-out.

Example:

```
404    program pltcalc                    (This calculation plots sentence 14
       call erase                          beginning at coordinates 70,180.
       ix = 70                             Sent14 has 22 characters.)
       iy = 180
       nchar = 22
       karlist(0) = -1
       call plot(ix,iy, karlist,nchar,sent14)
       end
       x
405    program comment
             .
             .
             .
```

The last calculation (the end of the entire source program) is terminated by an End, Finis, X, and a BCD end-of-file:

```
492    program name
c      this routine plots each character as it is pushed
c      to form the student's name on the screen
       inkount = inkount + 1
       if(inkount - 18) 1,2,2
1      call key(nkey)
       call keychar(nkey,karlist(1))
       ix = 70 + 7 ' (inkount - 1)
       iy = 96
       nchar = 1
       call plot(ix,iy,karlist,nchar)
       return
2      call ring
       end
       finis
       x
..
..
```

A calculation with arguments cannot be used as a calc by itself, but may be called from another calc. The placement of a calc with arguments in the list of calculations is arbitrary.

* No subroutine may be called in a function if the value of the function is to be returned to the calling program.

PLATO Format

The PLATO Format Statement

Form:

```
PLATO FORMAT ARRAY,NPERWORD,A(SPACE)SENTENCE(PERIOD)
```

Examples:

```
300   plato format sent1,8,welcome(kspace)to(kspace)plato(period)

302   plato format sent3,8,this(kspace)is(kspace)lesson(kspace)(n1)
     1(n4)(period)
```

The PLATO format statement packs character numbers into an array. The array name is automatically placed in internal common and may not appear in a DIMENSION or COMMON statement. Once defined, the array may be referenced by any calculation.

NPERWORD is the number of characters to be packed per computer word. If the total number of characters in the lesson 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 which form the sentence or message. If the names are only one letter long, like the alphabet characters in the examples above, they need not be separated from each other. If a name contains two or more letters, such as "kspace", it must be set off from the others by parentheses, (kspace).

The total number of computer words generated by a PLATO format statement is:

((no. of chars. in sentence)/NPERWORD + (1 if remainder, 0 if not) + 1)

The first word of a PLATO format array is a code word containing (NPERWORD-1) in the upper address and (shift per character) in the lower address.

PLATO format statements may be placed before the FORTRAN calculations or inside an individual calculation before the first use of the sentences and names they define.

Executable Logical Statements (to be used in calculations)

1. FORCE = NAME Causes the forced key to be that whose name is specified. In effect, this statement overrides the Force Key statement in the modeswitch. A forced key may be specified in a calculation even though a corresponding Force Key statement does not exist in the modeswitch.

2. IN(KEY,GROUP)YES,NO This statement is used to determine whether a key is contained in a given group. In calculations, group names may be used only within an In statement. The statement must be followed by two statement numbers. Control will go to the first if the key is in the group and to the second if it is not. The In statement is preceded by CALL KEY(I), a PLATO system routine:

```
Example:  call key(nkey)
          in(nkey,alpha)55,31
```

3. MODE = NAME Causes the next mode to be the mode whose name is specified. In effect, this overrides the Next Mode statement in the modeswitch for one time. No mode change may be made in an entry calculation, so assign a dummy key and force it inside the entry calculation:

```
Example:  mode two
          calc ntry        (The calc contains force = dmy.)
          keys dmy
          next mode four   (The mode change occurs here.)
```

4. NAME1 = NAME2 Causes the numerical value of NAME2 to be stored in the location of NAME1. NAME2 may be a key name, a character name, or a mode name. NAME1 may be a variable or array or student bank variable. If a key and a character have the same name, the key takes precedence. The corresponding character number can be obtained by CALL KEYCHAR(I,J), a PLATO system routine.

5. XINCRMT = N
   = XINCRMT*( )+( ) (any mathematical expression)

Statements of the first type above cause the X increment for base address used in plotting to be reset to the value of 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 in calculating starting coordinates, etc.

6.	YINCRMT = N	The Y increment for plotting. It may be used exactly like XINCRMT = N.
7.	XMAXMUM = N	The X maximum for plotting. It may be used exactly like XINCRMT = N.
8.	YMAXMUM = N	The Y maximum for plotting. It may be used exactly like XINCRMT = N.

PLATO System Routines (to be used in calculations)

1. CALL AUDIO(N) Selects audio effect N.

2. CALL BKSP(X,Y,N) Backspaces the plotting coordinates for base address N times. Tests for X or Y equal to 0. Resets X and Y back to XMAXMUM or YMAXMUM as required. Upon exit, X and Y contain the new coordinates, respectively. N remains unchanged.

3. CALL ERASE Completely erases the student's storage device. Upon exit, the storage device is set to "write."

4. CALL EQUIP1 Activates special equipment #1.
   CALL EQUIP2 " " " #2.
   CALL EQUIP3 " " " #3.

5. CALL INTRUPT Causes the calculation to be interrupted at this point to process requests from other students. When all other requests are processed, this routine automatically comes back to continue where the calculation was interrupted.

6. CALL KEY(I) Stores the number of a key being processed for a student in location I.

7. CALL KEYCHAR(I,J) Takes the key number stored in location I, find the number of the character whose name is the same as the key and stores the character number in location J. If there is no character of the same name, a 0 is stored in location J.

8. CALL NXTLINE(Y,N) Moves the Y coordinate for base address down N lines without changing X. If YMAXMUM is exceeded, YINCRMT is reset to 0 as required. Upon exit, Y contains the new coordinate and N remains unchanged.

9. CALL PACK(NPERWORD,PACKLIST,KARLIST,N) Packs N character numbers in array KARLIST, beginning at KARLIST(1), into the array PACKLIST, beginning at PACKLIST(1). NPERWORD is the number of characters from karlist to be packed into each word of PACKLIST. PACKLIST(0) contains (NPERWORD-1) in the upper address and (shift per character) in the lower address.

NOTE: The next two plot routines are essentially the same routine except for the number of arguments. If PLOT with 4 arguments precedes PLOT with 5 arguments in the <u>same</u> calculation, give the first PLOT statement a fifth (dummy) argument.

10. CALL PLOT(IX,IY,KARLIST,N) Plots N equential characters on the student's television screen beginning at IX,IY. When the series of plots is completed, N, KARLIST(0), IX and IY are altered:

N' = 0
KARLIST(0) = 0
IX' = IX + N * XINCRMT. Whenever IX' is greater than XMAXMUM, IX' is reset to 0. If the number of characters plotted after IX' is reset to 0 is M, IX' = 0 + M * XINCRMT.
IY' = IY + (no. of times IX' was reset to 0) * YINCRMT. Whenever IY' is greater than YMAXMUM, IY' is reset to 0. If the number of additional times IX' was reset to 0 is M, IY' = 0 + M * YINCRMT.

IX, IY, and N must be integer (fixed point) variables in the student bank. KARLIST must be an integer FORTRAN array of at least one dimension in the student bank.

If KARLIST(0) is not changed before calling PLOT, it is automatically 0 since all student bank variables are initially 0. PLOT sets KARLIST(0) to 0 each time a plot is completed. When KARLIST(0) is 0, karlist is assumed to contain right-justified numbers stored in reverse order to the direction of subscripting:

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

All locations of KARLIST are unchanged when the plot is completed and the storage device is set to "read."

11. CALL PLOT(IX,IY,KARLIST,N,PACKLIST) The addition of PACKLIST as an argument is essentially the combination of UNPACK and PLOT. If KARLIST(0) was set to -1 since the last plot for this student, the PACKLIST option will be selected. PACKLIST must be a FORTRAN array and must have been packed by a routine which places (NPERWORD-1) in the upper address and (shift per character) in the lower address of PACKLIST(0).

Subsequent words of PACKLIST must contain NPERWORD character numbers in each word, packed in sequence from left to right, with the entire word left-justified if there are any unused bits.

With the PACKLIST option, KARLIST(0) and KARLIST(1) are used for temporary storage during plotting. Upon exit, KARLIST(0) = 0, KARLIST(1) = last character plotted, and the storage device is set to "read."

12. CALL PLOTG(IX,IY,KARLIST,N)
    CALL PLOTG(IX,IY,KARLIST,N,PACKLIST) Plots exactly like the other PLOT routines, but does not automatically set the storage device to "read" when the plot is completed. The characters plotted by PLOTG are not visible on the television screen until the student's storage device is set to "read" either by a subsequent plotting routine or by CALL READ, another PLATO system routine.

13. CALL PRVLINE(Y,N) Moves the Y coordinate for base address up N lines without changing the X coordinate. Tests for Y = 0 and resets Y = YMAXMUM as required. Upon exit, Y contains the new coordinate and N remains unchanged.

14. CALL READ Sets the student's storage device to the "read" state.

15. CALL RING Flashes the error light on the student's keyset. The light is turned off when the next key is pushed. The only way the light can be turned off is by pushing the next key.

16. CALL SELRASE(X,Y,N) Selectively erases the storage device at N sequential character positions, starting at X,Y. When the series of selective erases is completed, X, Y, and N are altered:

    N' = 0
    X' = X + N * XINCRMT. Whenever X' is greater than XMAXMUM, X' is reset to 0. If the number of character positions erased after X' is reset to 0 is M, X' = 0 + M * XINCRMT.
    Y' = Y + (no. of times X' was reset to 0) * YINCRMT. Whenever Y' is greater than YMAXMUM, Y' is reset to 0. If the number of additional times X' was reset to 0 is M, Y' = 0 + M * YINCRMT.

17. CALL SLIDE(N) Selects the slide whose number is contained in location N of the student bank. If the slide number is always the same in a particular calculation, N may equal the slide number itself. N must be less than or equal to 121. N is modulo 128; presently 122 to 127 are blanks.

18. CALL SPACE(X,Y,N) Spaces the coordinates N times. Tests for XMAXMUM and YMAXMUM exceeded as PLOT does. X and Y are correctly altered upon exit. N is unchanged.

19. CALL UNPACK(PACKLIST,KARLIST,N)   Unpacks N character numbers of PACKLIST into KARLIST in reverse order, starting at KARLIST(N). Thus KARLIST is suitable for use with PLOT. PACKLIST must have been packed by a routine which places (NPERWORD - 1) in the upper address and (shift per character) in the lower address of PACKLIST(Ø), such as PACK or the PLATO Format statement.

20. CALL WRITE   Sets the student's storage tube to the write state.

<u>Other PLATO Systems Routines on the Master Tape</u> (to be used in calculations)

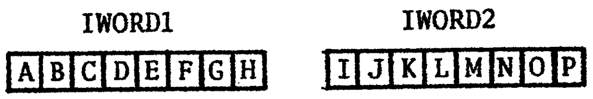
1. CALL CONNECT(N1,N2) This subroutine will create a fictitious input, according to the arguments given and place it in a proper place in the request list. N1 is the key name and N2 is the student number. It will appear as though student N2 pushed key N1, thus making intercommunication between student stations possible.

   Example: call connect (n4,12)

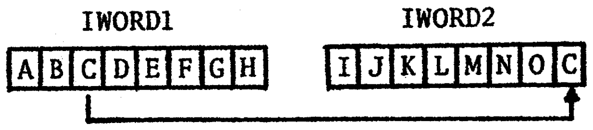
2. CALL GETPUT(IWORD1,IPOS1,IWORD2,IPOS2) This subroutine is a special case of TRANSFER. It puts the contents of character position IPOS1 of IWORD1 in position IPOS2 of IWORD2 without destroying the contents of the other character positions of IWORD2. Upon exit only IWORD2 is changed in any way.

   The subroutine treats each word as 8 six-bit characters numbered from the left.

   <u>Example</u>



   CALL GETPUT(IWORD1,3,IWORD2,8) would result in:



3. CALL GETPUT2(IWORD1,IPOS1,IWORD2,IPOS2) This subroutine, like GETPUT, is a special case of TRANSFER. It puts the contents of two adjacent characters beginning at IPOS1 of IWORD1 into two adjacent positions of IWORD2 beginning at position IPOS2. Upon exit only IWORD2 is changed in any way.

   The subroutine treats each word as 8 six-bit characters numbered from the left.

4. CALL RDBANK(IW1,ISTVAR,ISTNUM) This subroutine reads from a student bank. It places the contents of the student bank variable, ISTVAR, for student number,ISTNUM, into location IW1. ISTVAR <u>must</u> be a student bank variable.

5. CALL STBNK(IW1,ISTVAR,ISTNUM) This subroutine stores into a student bank. It places the contents of IW1 (which may be any type of constant or variable) into the student bank location, ISTVAR, of student number ISTNUM. ISTVAR <u>must</u> be a student bank variable.

6. CALL SET6(ISTUD) This subroutine allows one student to reference another student's bank. After a "CALL SET6(ISTUD)", all subsequent operations involving student bank variable names will use the values foudn in student ISTUD's bank until "CALL RESET6" or an external operation such as PLOT, ERASE, etc. is performed. After such an external operation, the program is in an <u>undetermined state</u>. Therefore, it is necessary to perform either a SET6(ISTUD) or a RESET6 if any further student bank references are made.

<u>Example</u>
```
CALL SET6(ISTUD)
II = 5
CALL PLOT(IX,IY,KARLIST,II)
```

The above sequence will plot the five characters from student bank ISTUD. If no further student bank operations are to be performed in the calculation, no reference to index register 6 has to be made. To be sure that each student is referencing his own bank a "CALL RESET6" must follow. To be sure that operations continue in the ISTUD bank, a "CALL SET6(ISTUD) is sufficient only if ISTUD is a constant or a non-student-bank variable. If ISTUD is a student bank variable, two statements are required:

```
CALL RESET6
CALL SET6(ISTUD)
```

This last is due to the fact that one cannot simultaneously reference two student banks using SET6.

7. CALL RESET6 This subroutine resets index register 6 (and hence all student bank operations) to the appropriate value for the student whose key is currently being processed.

8. CALL TRANSFR(LIST1,NBG,NCH,IQUIV,IGNORE,KOUNT,KOMPARE,KARFIND,NFINBG,LIST2)
This routine is a multipurpose character handling subroutine, based on 6-bit character numbers. It picks up a string of characters in an original list, LIST1, starting at a specified position in the list and transfers them to a final list,LIST2, subject to various tests.

TRANSFR has 10 arguments. The locations used by these arguments should be reserved by the Student Bank statement or by a DIMENSION statement:

LIST1	The original storage of (packed) characters.
NBG	The position to indicate where to start with reference to LIST1
NCH	The number of characters in LIST1 to be examined starting at NBG

Example:

kans								
				2	3	5	+	5
	.	6						

character positions with reference to kans:
1 through 8
9 through 16
17 through 24

```
With reference to kans(0):  nbg = 12
                            nch =  7
                            call transfr(kans,nbg,nch,...
With reference to kans(2):  nbg =  4
                            nch =  7
                            call transfr(kans(2),nbg,nch,...
```

IQUIV	Contains pairs of characters to be treated as equivalent
IGNORE	Contains characters to be ignored completely. These characters are not included in the NCH count.
KOUNT	Contains characters to be only counted in the NCH count.
KOMPARE	Contains characters to be tested against
KARFND	Contains the character, if any, found in KOMPARE
NFINBG	The position to indicate where to start with reference to LIST2
LIST2	The final storage for characters which pass the IGNORE. KOUNT, and KOMPARE tests, starting with NFINBG in packed form

LIST1, IGNORE, KOUNT, KOMPARE and LIST2 can be single word variables or arrays of any length. Pack 8 character numbers per word, left-justified. For IGNORE, KOUNT and KOMPARE, the termination character 00_8 must always follow.

IQUIV also can be a single word variable or an arry of any length. Pack 4 equivalent pairs of character numbers per word, left-justified. The termination character 00_8 must always follow.

LIST2 may be IQUIV, IGNORE, KOUNT or KOMPARE, that is to say the banks may be constructed as tests proceed.

MTY, a single word variable, must be reserved in COMMON, if a test is to be skipped. MTY is the dummy argument. Store a -0 (minus zero) in this location.

Calling Sequence for TRANSFR:

1. Set NBG, NCH and NFINBG. The values of these three arguments must be equal to or greater than 1. If NFINBG is 0 or negative upon entry to TRANSFR, the restoring process will be skipped.
2. Call TRANSFR with the arguments desired.
   a) LIST1, IQUIV, IGNORE, KOUNT and KOMPARE may be subscripted with any integer. If the location given contains 0, TRANSFR assumes that it is the beginning of an array and processing starts with the next word.
   b) LIST2 <u>must</u> be subscripted if it is an array, for characters will be stored in LIST2 starting at the location given in the calling sequence.
   c) If one wishes to skip various tests, write MTY in place of IQUIV, IGNORE, KOUNT, KOMPARE, KARFND and NFINBG when desired.

   <u>Example</u> `call transfr(listl,nbg,nch,mty,ignore,kount,mty,mty,mty)`

   This sequence of arguments will skip "equivalent", "compare" and "restore." The LIST2 argument may be dropped.
   d) If the first character in IQUIV, IGNORE, KOUNT or KOMPARE is 00_8, that test will be skipped.

Upon exit from TRANSFR:

1. If NBG or NCH were zero or negative upon entry to TRANSFR, it will exit immediately with NCH=0 and KARFND= -0 (error exit).
2. Characters found in IGNORE, KOUNT and KOMPARE are not restored in LIST2.
3. When the "compare" test is included the main FORTRAN program should check, upon returning, to see if KARFND=0.
   a) If a character in question is found in KOMPARE, exit from TRANSFR will occur with character found in KARFND and proper values in NBG,NCH and NFINBG. These are already incremented or decremented to be ready for the next character.
   b) If no character from LIST1 is found in KOMPARE, exit from TRANSFR will occur with KARFND=0, NCH=0. NBG AND NFINBG are incremented to be ready for the next character.
4. When the "compare" test is skipped, exit from TRANSFR occurs when NCH=0 NBG and NFINBG are incremented as if ready for the next character.

Remarks

1) STOP and PAUSE Statements are not allowed.

2) FORTRAN Machine Language may be employed when necessary. However, care must be taken to insert index register 6 wherever the operand address refers to one of the student bank variables.

3) The contents of location 0, which may be sampled at any time by the program, gives the elapsed time, in 60ths of a second, from the start of a CATO program.

C. PROGRAM COMPILATION AND EDITING

Compiling

To compile a PLATO program:

1. Put the PLATO Master Tape (#100) on tape unit 1.
2. Put a scratch tape on tape unit 2. The binary version of the program will be written on this unit if the program compiles.
3. Put the program source tape on unit 3 or 4.
4. Push the autoload button.
5. Type: cato,etc.; as described below.

CATO,I,L,O,P,S,M,E. This routine compiles a PLATO program and lists the PLATO source program and FORTRAN errors if desired. If the source tape is correct, a binary version of the program is written on the medium specified by argument O. If argument O is a zero, the binary version is written on tape unit 2 and an automatic CATORUN,NAME,2,1,P. puts the program in memory ready to run.

I = input medium (PLATO source tape)
L = FORTRAN library medium (PLATO master tape)
O = output medium for binary version of the program
P = source language output medium
S = symbolic modeswitch output medium
M = MAP (symbolic machine language) output medium
E = error listing medium

Example Type: cato,3,1,Ø,p,p,p,p.

3 = PLATO source tape on unit 3
1 = PLATO master tape on unit 1
Ø = binary version of the program written on tape unit 2 and read into memory ready to run
p = source language (BCD) output on printer
p = symbolic modeswitch output on printer
p = MAP language output on printer
p = error listing on printer

$\left.\begin{array}{l}\text{p = source language (BCD) output on printer}\\ \text{p = symbolic modeswitch output on printer}\\ \text{p = MAP language output on printer}\\ \text{p = error listing on printer}\end{array}\right\}$ complete printout of compilation

Errors

Error Messages During Compilation:

1. alphabet name needed - FATAL error. This message will appear immediately after the statement where the name is needed.

2. assignment interpreted modulo 128 - NON-FATAL error. The octal input number given in the assign statement is greater than 200_8.

3. coordinate oversized - FATAL error. A coordinate in the character statement is outside the 64 x 64 plotting area for a character.

4. directive out of order - FATAL error. A modeswitch statement precedes the first mode statement or some statement in a mode description is in the wrong place.

5. extra comma - NON-FATAL error. There is an extra comma somewhere in the statement.

6. illegal subroutine name used - FATAL error. A calculation name LISTARG, DELTARG or RING was used as a calc name.

7. (list) are circularly defined - FATAL error. This message refers to a set of modes. There is no basic mode in the references which is clearly and fully defined.

8. (list) are undefined modes - FATAL error. A mode name is given which ha corresponding mode in the modeswitch.

9. (list) were circular - FATAL error. This message refers to groups. There is no basic group in which all the keys are clearly and fully defined.

10. (list) were unassigned - FATAL error. Some group or groups are not fully defined by assigned keys.

11. (name) already given - NON-FATAL error. Duplicate names appear in one keys statement, mode statement, or calc statement.

12. (name) was already copied onto this mode - NON-FATAL error. The mode named is already part of the mode being processed.

13. (name) was reassigned - NON-FATAL error. A key name has been used twice in the assign statements.

14. (name) was reshaped - NON-FATAL error. Coordinates for one character name have been given twice.

15. no characters given - NON-FATAL error. A character statement with no coordinates exists.

16. numeric name needed - FATAL error. This message will appear after the st ment where the name is needed.

17. only one argument read - NON-FATAL error. The PLATO Compiler reads only one argument for Copy Mode, Next Mode, Same As, and Force Key statements. No stacking of arguments.

18. plato format name already used - FATAL error. A duplicate plato format name appears in a different plato format statement.

19. plato format variable name conflict - FATAL error. The plato format name used is also a character name. The conflict comes in plotting.

20. possible error - NON-FATAL error. All the calcs mentioned in the modeswitch do not have corresponding FORTRAN programs of the same name.

21. storage limit - FATAL error. The total storage is exceeded. The program is too large for the memory.

22. student variable previously specified - FATAL error. The variable name has appeared in a previous declarative statement.

23. student variable previously used - FATAL error. The student variable name appears before its declaration as a student variable.

24. tag overflow - FATAL error. The program is too large in a particular segment.

25. (type) format error - FATAL error. The syntax of the statement is incorrect (spelling, parentheses, comma, etc.).

26. undefined character name - FATAL error. A character name appears in a FORTRAN program which was never defined as a character.

27. unidentified logical key name - FATAL error. The key name appears before its declaration.

28. zero characters per word given - NON-FATAL error. NPERWORD in a plato format statement equals 0 or is not specified.

If two modes have the same name, replacement of statements within the final composite mode of that name is indefinite. Check the modeswitch printout for the actual composition. This error is NON-FATAL in compilation and is not tagged.

If the same key is listed twice in the same mode, replacement of the statements defining that key is indefinite. Check the modeswitch printout for the actual key definition. This is a NON-FATAL error and is not tagged.

Editing

EDIT,I,O,C. This routine corrects a BCD source tape; it is a FORTRAN resident service routine. EDIT is controlled by special directives from medium C which properly sequence insertions, deletions and substitutions as the contents of the unedited source tape I are transferred to a new medium O. The directives are:

	column 1	columns 2-5	column 6	columns 7-9
Insert	I	statement number or other tag	blank	up to 3 decimal digit numbers appearing anywhere in this field
Replace	R	"	"	"
Delete	D blank	" "	" "	" "
Move & Copy	M	columns 2-72, a complete record appearing on the source or blank for the last record of the edit tape		

Upon detecting a I, R, or D directive in column 1 of the edit tape on medium C, EDIT copies the old medium to the new medium searching for a record which contains a tag in columns 1-5 (2-5 if column 1 is non-numeric) which is identical to tag in columns 2-5 of the directive.

When this record is found, the decimal digits in columns 7-9 of the directive are interpreted as an integer N, and N more records are copied.

The last record read will be called the key record and is not copied immediately onto the new medium O, but is held in storage. Spaces a completely ignored.

The directive then positions the input medium I at the proper place for editing, and controls what sort of editing is to be done.

A directive record with spaces in columns 2-5 will hunt for the first record on the old medium I which has no tag in columns 1-5, or possibly a record with only a non-numeric character in column 1.

The I Directive — The records following this directive on the edit tape are inserted in the source program after the key record. Termination is by detecting a new directive.

The R Directive — The record following the directive replaces the key record. Records following replace those following the key record on a one-to-one basis. Termination is by detecting a new directive.

The D Directive

The record following the directive is read and interpreted in the same manner as the directive. Records between these two key records and including these two key records are deleted. The second record must have a space in column 1, and must be followed by another directive. At least two records are deleted by this directive

The M Directive

The M directive has a slightly different format. Its effect is to copy the old medium I onto the new med um O searching for a record identical (except for spaces) to that in columns 2-72 of the directive. Column 1 of the source medium is included in the search. The key record thus found is the first one examined by any of the other directives.

It is expected that the M directive will generally be used to position the input medium I at the start of a given program or subroutine.

A special M directive which is blank except for an M in column 1 is used as the last record on the edit tape. Upon identifying this directive, EDIT copies the input medium onto the output until two consecutive records containing period codes, "..", in columns 1 and 2 are found, causing normal exiting from the routine.

NOTE: All corrections must be arranged in the order in which the statement numbers they refer to appear on the source tape on medium I. Editing goes forward through the source tape, never backward.

Error Exits from EDIT:

Any messages typed on the typewriter indicate that errors have been detected.

1. no directive - No directive letter was found in column 1 when one was needed.

2. ../.. detected - The end of the source tape was reached before a blank M directive was given.

3. format col 7-9 - A character other than a numeral appears in columns 7-9 of the present directive.

4. d format error - The second record in a D directive has something other than a space in column 1.

Jump Switch Option for EDIT:

Setting J-1 will cause replacement of columns 73-120 of all records from the old source medium I with blanks (spaces) when they are transferred to medium O.

Other Uses of EDIT:

EDIT may be used for purposes other than correcting a source program. For instance, it may be used to position a source medium at a given record preparatory to some other operation. In such a case the control program would be:

```
edit,source medium,x,correction medium

m  (a record sought for)
i     x(col. 7) - a deliberate error to cause exiting from EDIT

bksp, source medium - to position the key record so it is ready to
                      be read again (possible only for magnetic tape).
```

FORTRAN RESIDENT

The FORTRAN resident handles the tape operations, typewriter, and printer control, punch and reader control -- all input and output operations required to create a complete and correct source tape for a lesson.

A low-pitched tone is heard when the computer is waiting to accept instructions from the typewriter. All typewriter and paper tape inputs must be in lower case.

A high-pitched tone is heard when the computer is waiting to punch or to read paper tape. When a seventh-level is read from the paper tape, the reader stops and the high-pitched tone is heard. Depressing the reader character-mode switch restarts the reader.

Two alternating tones are heard when the computer is waiting to print on the line printer. Check the printer-ready button or the CSX-1604 switch.

On paper tape input and output, a flex code Ø2, (question mark or underline) acts as a tab to column 73.

Autoloading with jump-switch 1 set automatically calls FORTBIN. Autoloading with jump-switch 2 set automatically calls CATOCOM, so that it does not have to be called by the CATO driver.

Stepping, clearing the console and setting the program address to $1Ø_8$ returns the control to the console typewriter. Setting the program address to $2Ø_8$ has the same effect as autoloading.

Standard Argument List (for use in FORTRAN resident subroutines)

t 9	- typewriter
c f	- paper tape
1-8	- the eight magnetic tape units
p	- printer
q	- printer, no line format control
m	- memory (a subroutine with m as an argument must be preceded by the subroutine SET.)
x	- dummy argument

<u>FORTRAN Resident Service Routines</u> (available without calling)

1. CATO,I,L,O,P,S,M,E. This routine is the driver for compiling a PLATO source program. For users' instructions, see Part III-C, Compiling.

2. CATORUN,I,L,X. This routine reads a PLATO binary program tape into memory ready to run. For users' instructions, see Part II-B, Read-Ins.

3. BKSP,N. Backspaces the tape unit specified by argument N.

4. CALL,I,NAME,NAME,... . Reads up to 10 binary programs into memory from medium I. If argument I is f, the reader must be in the assembly mode. The programs named are read into unique storage but are <u>not</u> executed. They must be called in the usual fashion.

5. CLEAR. Clears all of the memory above the resident.

6. CONTROL,I,O,E. Transfers control from the typewriter to medium I. Control statements are listed on medium O. Errors are listed on medium E.

7. COPY,I,O. Copies from one medium to another until a BCD end-of-file (..) is read and copied. When argument I is f, the paper tape must be in flex code (BCD). At the end of copying, the typewriter types the number of entries. Each entry is a 120-character block.

8. COPYR,I,O. Copies one record from medium I to medium O.

9. COPYS,I,O,C. Copies medium I to medium O comparing columns 1-5 of I with columns 1-5 of the directive from medium C. Spaces are ignored When an identical record is found, the number in columns 7-9 of the directive of medium C is interpreted and that many more records are read from I. All records but the last one are copied onto O. If argument I is magnetic tape, a backspace over the last record read occurs.

<u>Example</u>	Type :	copys,3,4,t. 208 5	Copies tape 3 to tape 4 until statement 208 is read from tape 3. Five more records are read from tape 3; the first four are copied to tape 4 and tape 3 is backspaced over the last record read. Tapes 3 and 4 are now positioned at the beginning of the fifth record past statement 208.
<u>Example</u>	Type :	copys,t,3,t.	Copies the typewriter input to tape 3 until ".." is read from the typewriter. A space followed by a carriage return will erase the current record typed if an error is made.

10. CP,I,O. Copies from one medium to another with the added verifying feature. Executes an automatic (REWIND,I, REWIND,O.)

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

11. DUMPS,A1,A2. Dumps the memory onto the line printer in octal. A1 is the beginning address in octal and A2 is the final address in octal. This routine is particularly useful for dumping the memory when the program occupies the entire memory. With this routine, there is no need to re-load the program. For users' information, see Part III-D, The Dumps Routine.

12. EDIT,I,O,C. This routine corrects a BCD source tape. It is controlled by special directives from medium C which properly sequence insertions, deletions and substitutions as the contents of the unedited source tape are transferred to a new medium O. For users' information, see Part III-C, Editing.

13. EFMARK,N. This routine puts a physical end-of-file mark on the tape specified by the argument number. (a physical end-of-file is a flip-flop set inside the tape unit.)

14. INPUT,I1,I2. Switches input from the medium specified in the program to another medium. Use the standard argument list to specify I1. Permissible designators for I2 are:

01 - typewriter	21 - magnetic tape unit 1
02 - reader, punch	22 - " " " 2
04 - line-printer (no line format control)	23 - " " " 3
	24 - " " " 4
05 - line-printer	31 - " " " 5
06 - memory	32 - " " " 6
00 - empty media (discard)	33 - " " " 7
	34 - " " " 8

Example Type: input,f,01. Switches the input from the reader to the typewriter.

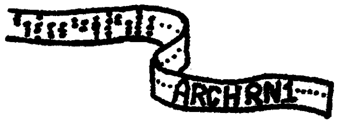
15. LOCK,N. Rewinds and unloads the tape specified by N.
LOCK,X1234. Rewinds and unloads all four tape units.

16. OUTPUT,O1,O2. This routine switches the output from the medium specified in the program to another output medium. Use the standard argument list to specify O1. Permissible designators for O2 are the same as those listed under INPUT,I1,I2.

Example Type: output,p,24. Switches the output from the printer to tape unit 4.

17. PTMAR,NAME. This routine punches up to eight alphanumeric characters on paper tape for visual identification. The first character must be a letter.

<u>Example</u> Type: ptmar,archrn1



PTMAR. Punches 2" of blank tape.

18. RESET. This routine resets the input and output media, changed by INPUT,I1,I2 and OUTPUT,O1,O2, back to the original media specified within the program.

19. REWIND,N. Rewinds the tape specified by argument N.
REWIND,F. Punches 6" of blank tape.
REWIND,M. Resets the memory pointer to the end of CATORES.
REWIND,P. Ejects a printer page.
REWIND,XNFMP. Performs all the above.

20. RUN.
RN. These routines execute binary FORTRAN programs.

21. SAVE. This routine causes the present memory bank to be saved if further programs are called.

22. SET,M. This routine allows the use of memory as a tape unit. M must be a memory address given in octal. The memory length is set to 0. This routine must precede any resident routine with m as an argument.

23. SKIP,I,NAME. Skips tape I until a record beginning with "..NAME" is read.
SKIP,X. Skips past the first record beginning with "..".

24. TRANSFER,I,O,NAME. This routine copies binary programs from tape I to tape O up to but not including the named program. By using argument x, the tape may be positioned without copying.

<u>Example</u> Type: transfer,1,x,endfile. Moves tape 1 to the end-of-file ready to add programs.

<u>Example</u> Type: transfer,1,x,name. Moves tape 1 to the beginning of the named program.

<u>Example</u> Type: transfer,4,f. Copies binary tape 4 to the punch.

25. VERIFY,I,O. Compares tape I with tape O. Tapes must be in BCD format. Verification continues until ".." is read. The typewriter prints the number of entries (120-character blocks) that have been verified. Error printout will give the block number after which the error occurred and a listing of the two blocks involved. Verification will continue after error printout.

26. VFYLIB,I,O. Compares binary tape I with binary tape O, counting the pairs of blocks compared. Error printout is the number of the pair of blocks containing the error.

D. CODECHECKING INFORMATION

Running a Program

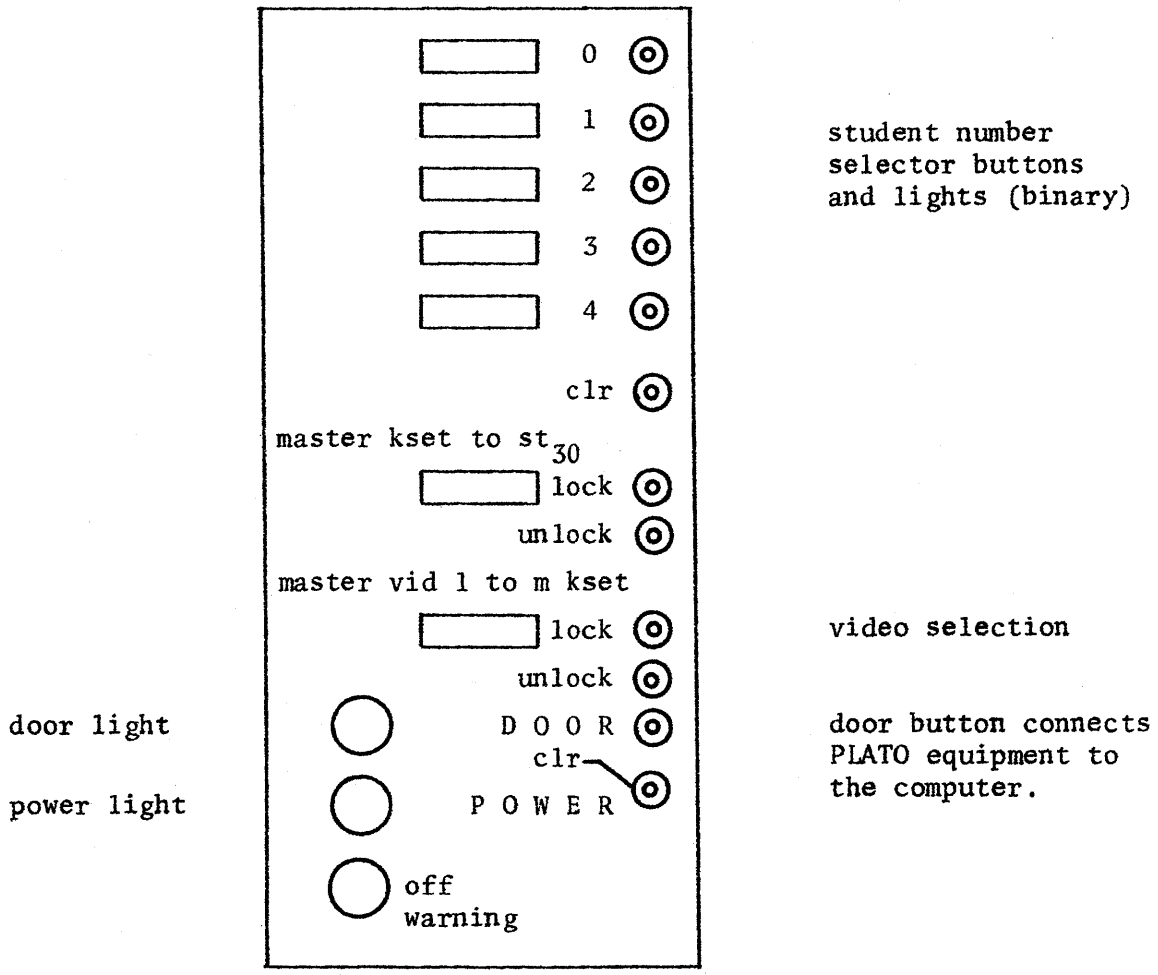
After the program is compiled, it is necessary to run the lesson to test whether the FORTRAN programs perform the intended functions with the PLATO equipment and to be sure that all the alternative moves associated with the teaching rules have been included and provided for throughout the program. To accomplish this, the programmer and/or teacher of the lesson need to put themselves in the position of a student operating the keyset and viewing the lesson material.

Follow the directions given in Part II-B on reading in a program as a student run. Be sure to test all the segments of the logic thoroughly.

Master Keyset Operation at the Console

There is a PLATO master keyset and television screen in the computer room which will allow the programmer to run his program while at the computer console. It is convenient for pushing keys and checking program operation simultaneously in the debugging process of developing a lesson.

The master keyset is equipped with a set of buttons for selecting any student stations for operation or viewing by the programmer, author, or teacher of a lesson. Manual selection of a student number (in binary) may be made at any time by pushing the buttons for that number. The master keyset selector controls are diagramed below:



Diagnostic J-2 Doping

Running a program with J-2 set, results in a printout of the keys pushed and the calculations performed during a trial diagnostic run. The printout also gives the sequence of mode changes which occur with each key input. Putting J-2 up during execution causes an on-line printout to occur in the following form:

```
STUDENT     KEY     MODE     CALC     (in octal)
  000       023      005      021
  002       012      001      004
   .         .        .        .
   .         .        .        .
   .         .        .        .
```

The printout of DOPE 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 will occur on the line printer.

Examining the Contents of Specified Memory Locations

It is possible to print the contents of specified memory locations in a program whenever desired and to continue on with the program where it was interrupted; i.e., with undisturbed "common", "student bank", and related banks such as "modebank", "justdone", etc.

To accomplish this:

1. Step. Master clear up and down.
2. Set the "program address" to 10 for typewriter control. Start.
3. Print the contents of the desired locations by using the "dumps routine, as described below.
4. Clear. Set the "program address" to 4201 (the entry to RESTART).
5. Start. When the channel 5 light comes on, continue with the run.

NOTE: If the program has been stopped due to programming errors, it may or may not be possible to continue by RESTART depending on the nature of the errors.

The DUMPS Routine

DUMPS,A1,A2. Dumps the memory onto the line-printer in octal. A1 is the beginning address in octal and A2 is the final address in octal. This routine is particularly useful for dumping the memory when the program occupies the entire memory. There is no need to reload the program. The format of the printout is similar to that of the ILLRES routine, SNAPSHOT, except:

a) It will print out a line of stars if the contents of one or more full lines are the same as the last word printed and the first word of the next line printed.

b) It will print out a line of dots after printing the last line.

For information about CATORES tables and constants (contents, beginning addresses, etc.), see Appendix D.

Making an Autoloadable Binary Tape

A. Starting with a source program tape:

1. Put the PLATO Master tape (#100) on tape unit 1, the source program on unit 3, and a scratch tape on unit 2.
2. Put the tape for the autoloadable binary on unit 4. (ring in)
3. Type: cato,3,1,0. (The program will be compiled and read into memory.)
4. When the console lights, push the door button and Start.
5. Step. Clear up and down.
6. Set the "program address" to 04203 (entry to PROGSAVE). Set the "accumulator" to 1.
7. Start. The autoloadable binary will be written on tape unit 4.

B. Starting with a binary program tape:

1. Put the PLATO Master tape on tape unit 1, the binary program tape on unit 2.
2. Put the tape for the autoloadable binary on unit 4. (ring in)
3. Type: catorun,2,1,x. (The program will be read into memory.)
4. Follow steps 4 through 7 above.

Drivers (CATORES)

1. CATORES - 04000. Resets all necessary tables, clears student bank and common, and begins the execution of the program currently in memory.

   Accumulator options for CATORES:

ACC = 0	J-2 printout occurs on the console typewriter.
ACC = (non-0)	J-2 printout occurs on the line-printer.

2. ENDOPE - 04200. The program to end a dope tape on tape unit 2. This causes all the necessary special records (end-of-session, end-of-dope-tape, and end-of-file) to be written. This program stops ready to jump immediately to the RESTART program. For more information, see Part II-B, Emergency Procedures.

3. RESTART - 04201. The program for the continuation of a run which has been interrupted. The dope tape must be on tape unit 2 and must have an end-of-dope-tape mark after the session to be restarted. The "student tables" (student bank, mode bank, justdone,...) and "common" are not reset. The clock is reset. Upon entry to RESTART, the jump switch settings are checked. For more information about the use of RESTART, see Part II-B, Read-Ins.

4. SPECTRE - 04202. The simulation program executing transcribed reruns of PLATO sessions. This program uses DOPE on tape unit 2 to simulate doped inputs. Tape 2 is never rewound. 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 outputs to the PLATO equipment. For more information about the use of this program, see Part II-B, Reviewing a Student Run Using SPECTRE.

5. PROGSAVE - 04203. The program to write memory onto tape unit 4 in the form of an autoloadable binary version of the lesson. If J-2 is set, tape 4 is not rewound. The FORTRAN resident, as well as the lesson program, is saved.

   Accumulator options for PROGSAVE:

ACC = 0	When autoloaded, the program stops ready to start at CATORES.
ACC = 1	...starts at RESTART (does not reset student tables).
ACC = 2	...starts at SPECTRE.

   For more information about the use of PROGSAVE, see Part II-B, Read-Ins.

A. JUMP SWITCH SETTINGS

No Jump Switches

If no jump switches are set, normal operation of the run will proceed without doping. The typewriter types: DOPING SUPPRESSED. The run then proceeds automatically.

Setting J-2

If J-2 is set until the channel 5 active light comes on, the program will rewind tape 2 before starting to write DOPE, if doping option J-1 or J-3 has been chosen.

If J-2 is not set before the channel 5 active light comes on, and either J-1 or J-3 is set, the program will skip to the end of the DOPE tape on unit 2, which is assumed to be a correctly terminated old DOPE tape containing DOPE from previous runs. This option allows the collection of DOPE from several runs on a single tape. The tape may run away if it was not previously terminated correctly.

The J-1 Option

If J-1 is set during entry and kept set during execution, the DOPE is written on tape unit 2, which must be ready to be written upon when the pr gram begins.

The typewriter types: TYPE MONTH/DAY/YEAR/ID. The operator must then type the tag by which the run is to be identified. 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 a typing error, may be restarted after a carriage return.

Examples:	2/22/44.	ju/ne/8.
	a.	1/1/65/7.
	11/28/64/a.	///q.
	te/st/.	

The J-1 option gives a record of input time of the keys, a record of the mode at the time each key is processed. J-1 gives no record of fictitious keys such as force keys.

Unsetting J-1 during execution causes the DOPE tape to be terminated by an end-of-session mark, an end-of-dope-tape mark and a physical end-of-file. The computer stops ready to execute the RESTART program. The same results can be achieved in the case of accidental stopping of the program by startin at 4200b (the entry to ENDOPE).

For more information on the use of J-1, see Part II-B, Read-Ins.

The J-3 Option

If J-3 is set during entry and kept set during execution, the DOPE is written on tape unit 2, which must be ready to be written upon when the program begins.

The DOPE written with J-3 set has a slightly different form from that written with J-1 set in that the key numbers for fictitious keys have been increased by 400b. This does not affect the validity of the data because the largest possible key number is 177b.

The J-3 option gives a record of the mode at the time each key is processed, a record of input time of the keys. J-3 also records all fictitious keys as real inputs increased by 400b.

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

B. DOPE IDENTIFICATION AND FORMAT ON MAGNETIC TAPE

The DOPE for all students simultaneously is written on tape unit 2 in 120-character BCD records. In each 120-character record, there are seven 16-character blocks and a final 8-character block. Each 16-character block is written as follows:

S	S	K	K	K	M	M	M	T	T	T	T	T	T	T	T

2 chars. = octal student number
3 chars. = octal key number (J-1 set = true number)
(J-3 set = number + 400b)
3 chars. = octal mode number (J-1 set = mode at the time of input)
(J-3 set = mode at the time of execution)
8 chars. = octal time in 15ths of a second (The time recorded is input time.)

The final 8-character block is in FORTRAN (A8) format, apportioned as follows:

7 chars. = the identification tag, up to seven characters in length, that was typed by the operator

11/28/64/a.	results in	112864a
a.	results in	∅a(followed by 5 BCD blanks)
1/1/64/7.	results in	∅1∅1647
te/st/s.	results in	test∅s(followed by 1 BCD blank)
//a.	results in	∅∅∅∅a(followed by 1 BCD blank)
///z.	results in	∅∅∅∅∅∅z
g//un.	results in	∅g∅∅un(followed by 1 BCD blank)
.	results in	∅∅(followed by 5 BCD blanks)

1 char. = 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.)

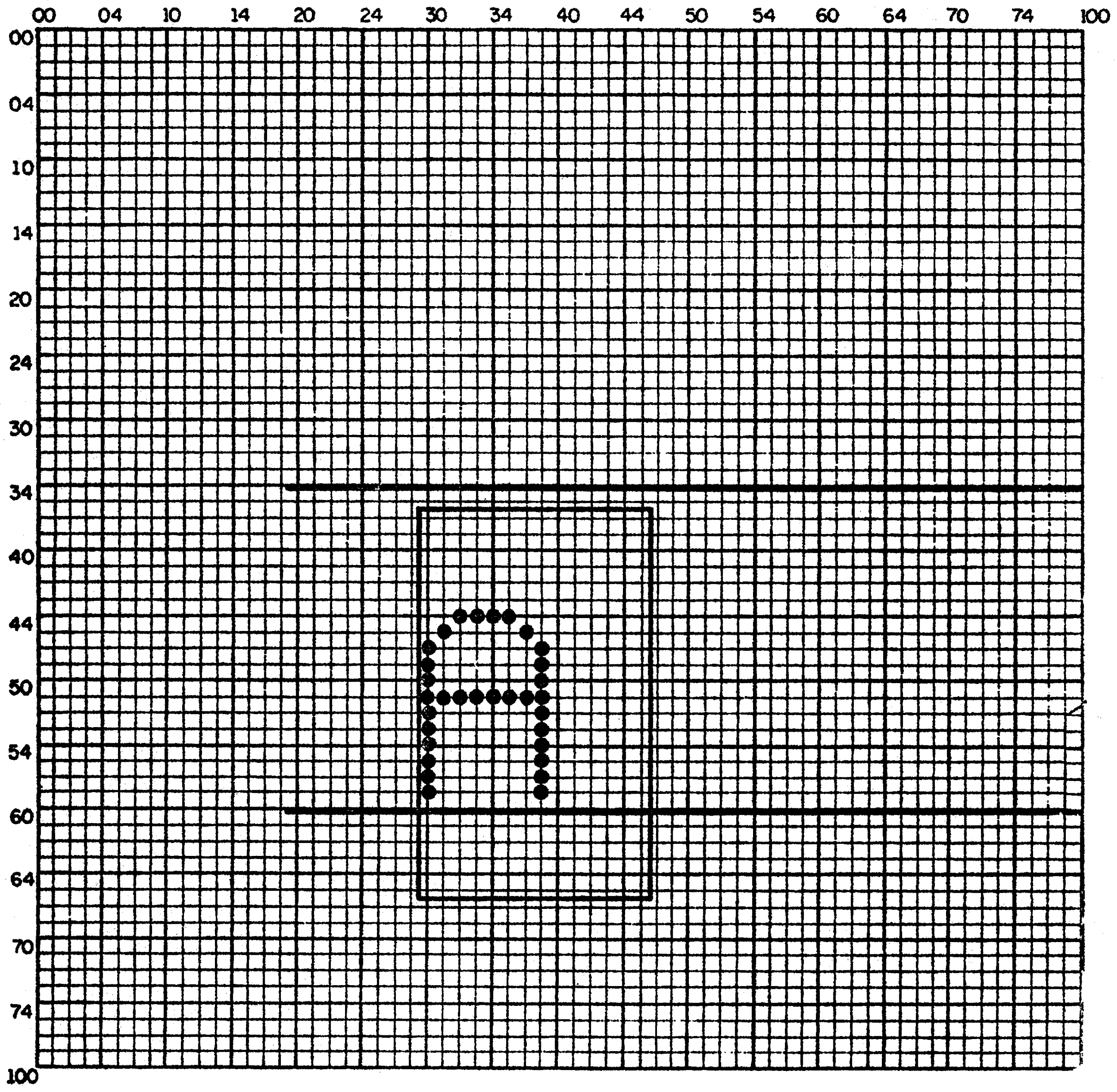
The clock is never ∅ except in a 16-character block which contains no information, so DOPE processing routines should ignore records with time ∅. If a restart (use of the RESTART program) occurs, the clock is reset to 1. Thus, a negative time change indicates that a restart occurred. A message must be typed.

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 the copying of this tape to a longer tape. Backspace over the physical end-of-file on the tape, and copy the remaining DOPE (assumed to have been recorded on a blank tape after the end-of-tape stop and subsequent continuation of the program) on to the end of the longer tape.

C. DOPE SORTING AND ANALYSIS

1. DPLIST1 - This routine prints out key inputs by name vertically on the printout page. It will print the entire tape of the number of individual sessions desired. For users' instructions, see Part II-B, Doping.

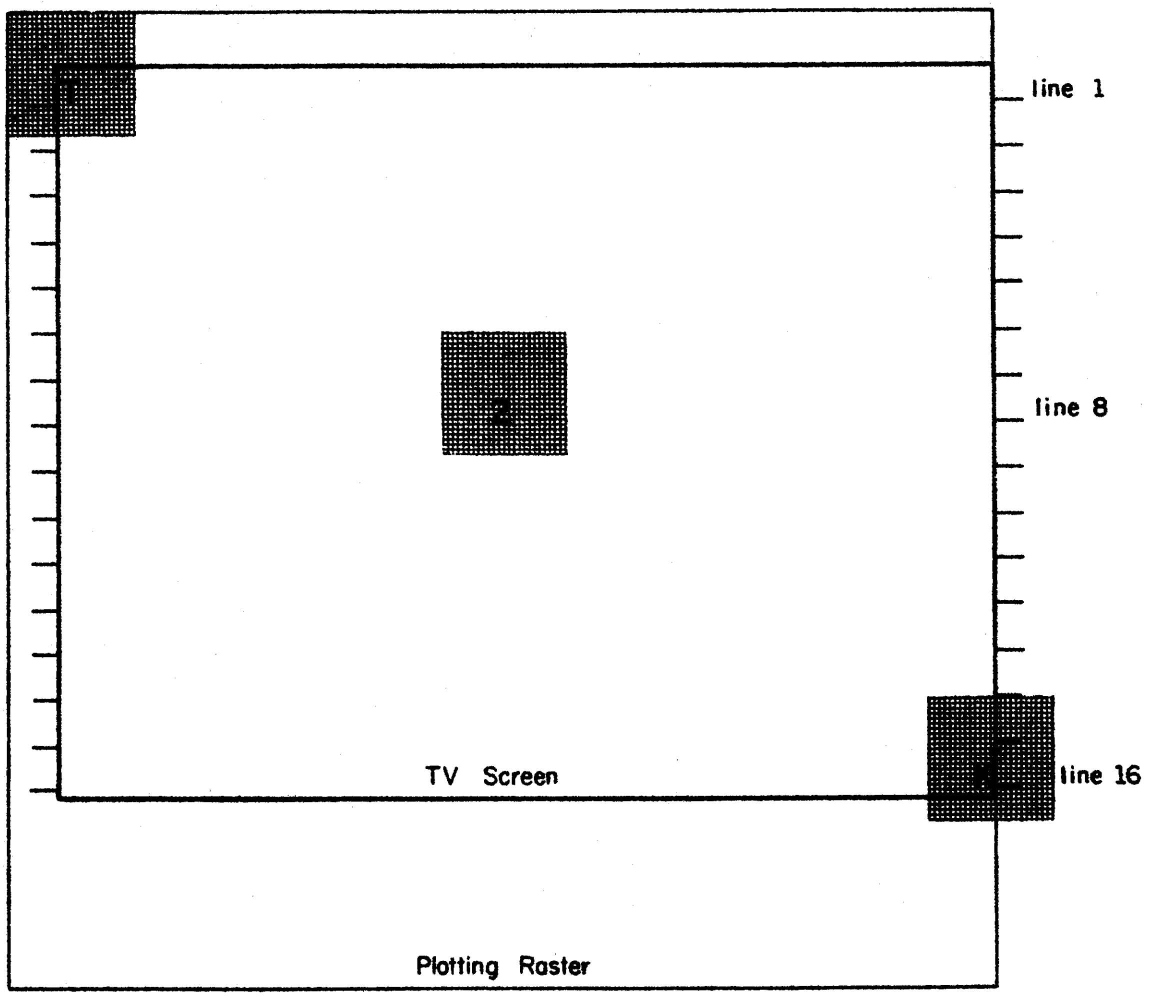
2. COPY,I,O - This routine will dump BCD records of DOPE onto the medium specified. For information on the use of this routine, see Part III-C, FORTRAN Resident service routines.



Characters to be plotted by a PLATO program are originally drawn as a series of dots (points) within a 64 x 64 point plotting square. The points which form the character are then translated into coordinate pairs which make up the bulk of a Character statement in a PLATO source program. The "A" above, in Character statement format is as follows:

```
      character a((30,57)(30,56)(30,55)(30,54)(30,53)(30,52)(30,51)
     1(30,50)(30,47)(30,46)(31,45)(32,44)(33,44)(34,44)(35,44)(35,45)
     2(37,46)(37,47)(37,50)(37,51)(37,52)(37,53)(37,54)(37,55)(37,56)
     3(37,57)(31,51)(32,51)(33,51)(34,51)(35,51)(36,51))
```

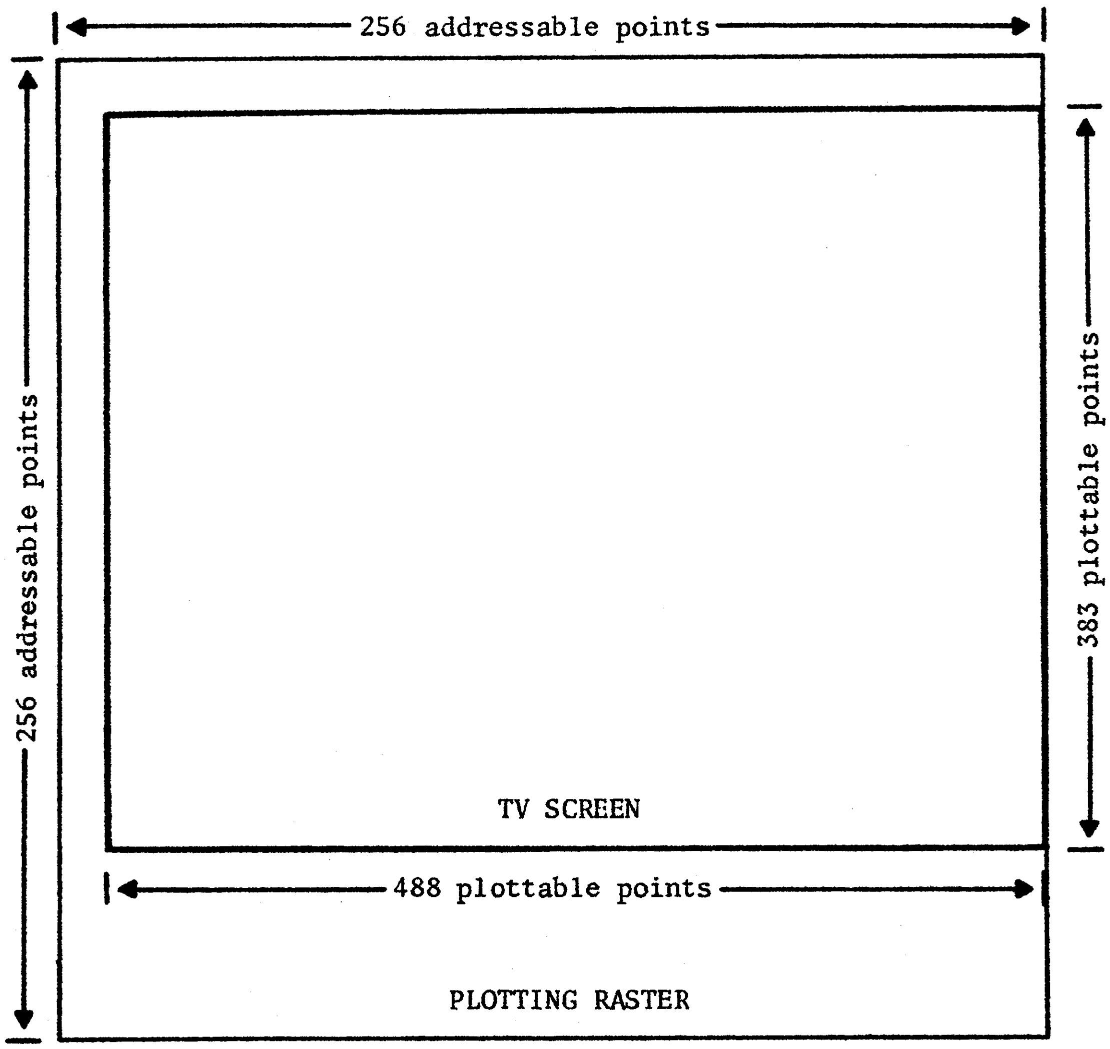
The number of points which constitute a character is limited to 9 continuation statements, about 88 points with a one to six letter character name. Each named character must fit within the plotting square. If larger symbols are desired, they must be plotted as several characters or as a single point with a changing base address.



The diagram above shows the size of a single plotting square in relation to the size of the television screen and the plotting raster. By convention, the characters are plotted in the same position on the single plotting square. This allows a standard chart for base address to be used in specifying the location of the square on the television screen. The characters shown above are slightly larger than Pica type.

Character T is position 1 on line 1 - Base Address 0,0.
Character 2 is position 16 on line 8 - Base Address 105,84.
Character K is position 35 on line 16 - Base Address 238,180.

The base address always refers to the upper left-hand corner of the single plotting square.


256 addressable points
256 addressable points
383 plottable points
TV SCREEN
488 plottable points
PLOTTING RASTER

PLATO III - Base Addresses for Character Plotting (6x9)

x coordinate			y coordinate	
			E.E. 322	Texttestor-ArithDrill
Char 1 -	0	Line 1 -	6	0
2 -	6	2 -	16	10
3 -	12	3 -	26	20
4 -	18	4 -	36	30
5 -	24	5 -	46	40
6 -	30	6 -	56	50
7 -	36	7 -	66	60
8 -	42	8 -	76	70
9 -	48	9 -	86	80
10 -	54	10 -	96	90
11 -	60	11 -	106	100
12 -	66	12 -	116	110
13 -	72	13 -	126	120
14 -	78	14 -	136	130
15 -	84	15 -	146	140
16 -	90	16 -	156	150
17 -	96	17 -	166	160
18 -	102	18 -	176	170
19 -	108			
20 -	114			
21 -	120			
22 -	126			
23 -	132			
24 -	138			
25 -	144			
26 -	150			
27 -	156			
28 -	162			
29 -	168			
30 -	174			
31 -	180			
32 -	186			
33 -	192			
34 -	198			
35 -	204			
36 -	210			
37 -	216			
38 -	222			
39 -	228			
40 -	234			
(41)-	240			

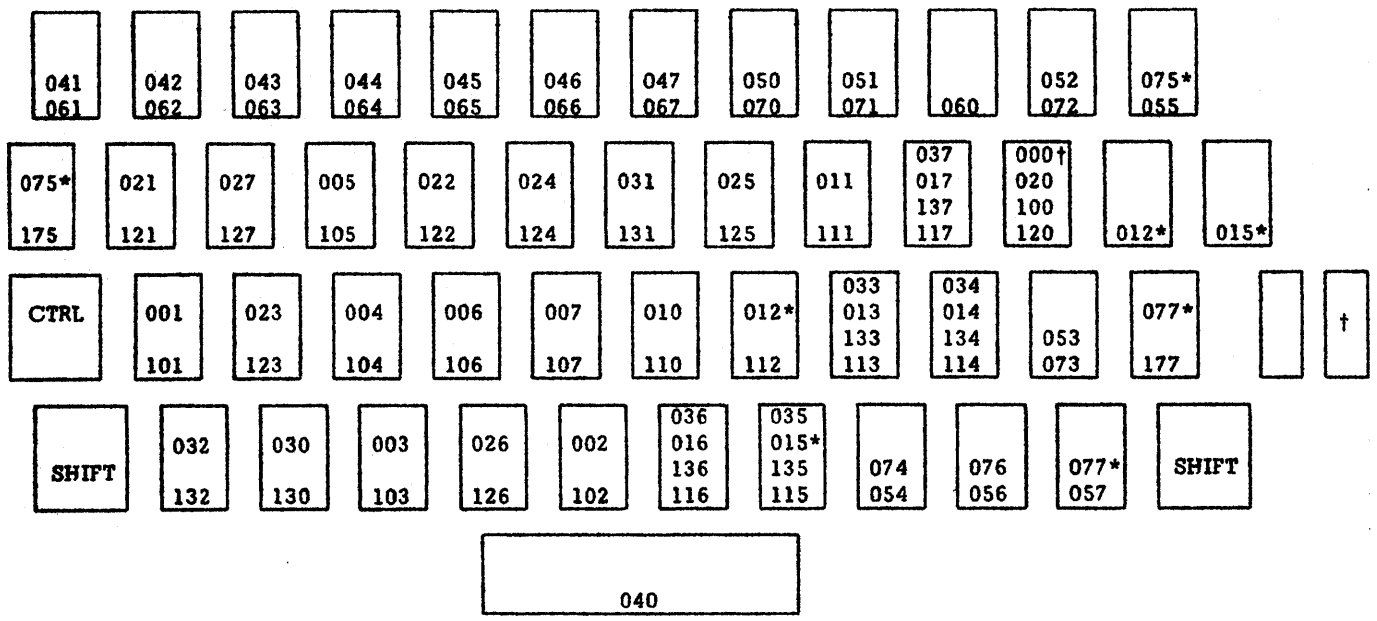
Note: The y coordinates above are compatible with the Selectric Typewriter.

The x coordinates will have to be approximated from the following:

Selectric Typewriter = 47 typed characters

Plotted Characters = 40(41) positions

The Textestor-ArithDrill y coordinates are compatible with selective erase, given the same coordinates.



*duplications:	012 - ctrl and normal
	015 - ctrl and normal
	075 - ctrl and shift
	077 - ctrl and shift

ctrl & shift
ctrl
shift
normal
- guide to numbering scheme

† input of all keys when break key is depressed

THE PLATO TELETYPE KEYSET

The PLATO keyset is a teletype keyboard which sends inputs to the computer when the keys are pushed. Each key has at least one input, and some have as many as four depending on what state the keyset is in. There are four states in all:

1) The normal state in which each key pushed singly sends an input to the computer
2) The shift state when the "shift" button is held down as a key is pushed causing a different input from normal to be sent to the computer
3) The control state when the "control" button is held down as a key is pushed sending still another different input to the computer
4) The control and shift state in which both the "control" and the "shift" buttons are held down as a key is pushed to send a fourth input different from all the others to the computer.

There are 101 inputs altogether. Ninety-six of them are unique; four are duplications, and input 000 is the input from every key when the break key is held down. The break key disengages all the keys and is commonly used to "block out" inputs which would disrupt the program as keycaps are being attached to the keyset.

The programmer may use as many keys as needed in any combination which suits the purposes of the lesson. A key may have many different functions throughout the program and may belong to a variety of key groups.

The inputs are given symbolic names within the computer program and are referred to by name throughout the program. See Part III on PLATO Statements - the Assign statement and the Group statement.

CATORES (The Resident Program for Execution of PLATO Programs)

When CATORES is entered, it checks first to see if the PLATO equipment is connected to the computer. A typeout occurs if it is not. It then 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 banks are then set to Ø, the request list (REQLIST) is set to -1 to indicate that no information is present, JUSTDONE and MODEBANK are set to Ø, NOCALC is made the Øth 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 Ø is selected. The clock is set to 1/15 of a second and started. The clock is increased by 1 after each 1/60 of a second. Channel 5, the PLATO input channel, is then activated and the computer waits for a student to press a key.

The initial mode is made Ø. The entry calculation, 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 Ø returns to mode Ø, the entry calculation will be performed when the next mode is set to Ø, before the second key is pushed.

If a PLATO systems output subroutine 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 calculation may possibly be used by another student before one student has completed it. When all requests waiting from 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.

CATORES contains all the PLATO systems routines listed in Part III. If a subroutine listed on the PLATO master tape is called in a calculation, it is read from the master tape during compilation and becomes part of the binary version of the program.

CATORES TABLES & CONSTANTS

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

2. CALCLIST 128 words. Gives the addresses of CALCS Ø through 255, packed two to a word. Currently, only 254 calculations may be defined by a PLATO 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 a PLATO program. It is divided into equal blocks, one for each mode. Each mode is divided into sections; each section contains the calculation number, the force key number, and the next mode number required by a specific key.

5. LENGTH 1 word. Gives the number of words required 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 equals the internal character number corresponding to the nth internal key number. Lower address equals the internal key number corresponding to the nth physical key number.

8. PLOTABLE N words. Upper address equals the number of points in a given character. Lower address equals the relative position in the 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 directive begins a new word.

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

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

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

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

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

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

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

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 equals the internal key number corresponding to the physical key pushed. The lower address equals the student number. A partially executed request has the second bit from the left set. If the list becomes full (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 state.
   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.

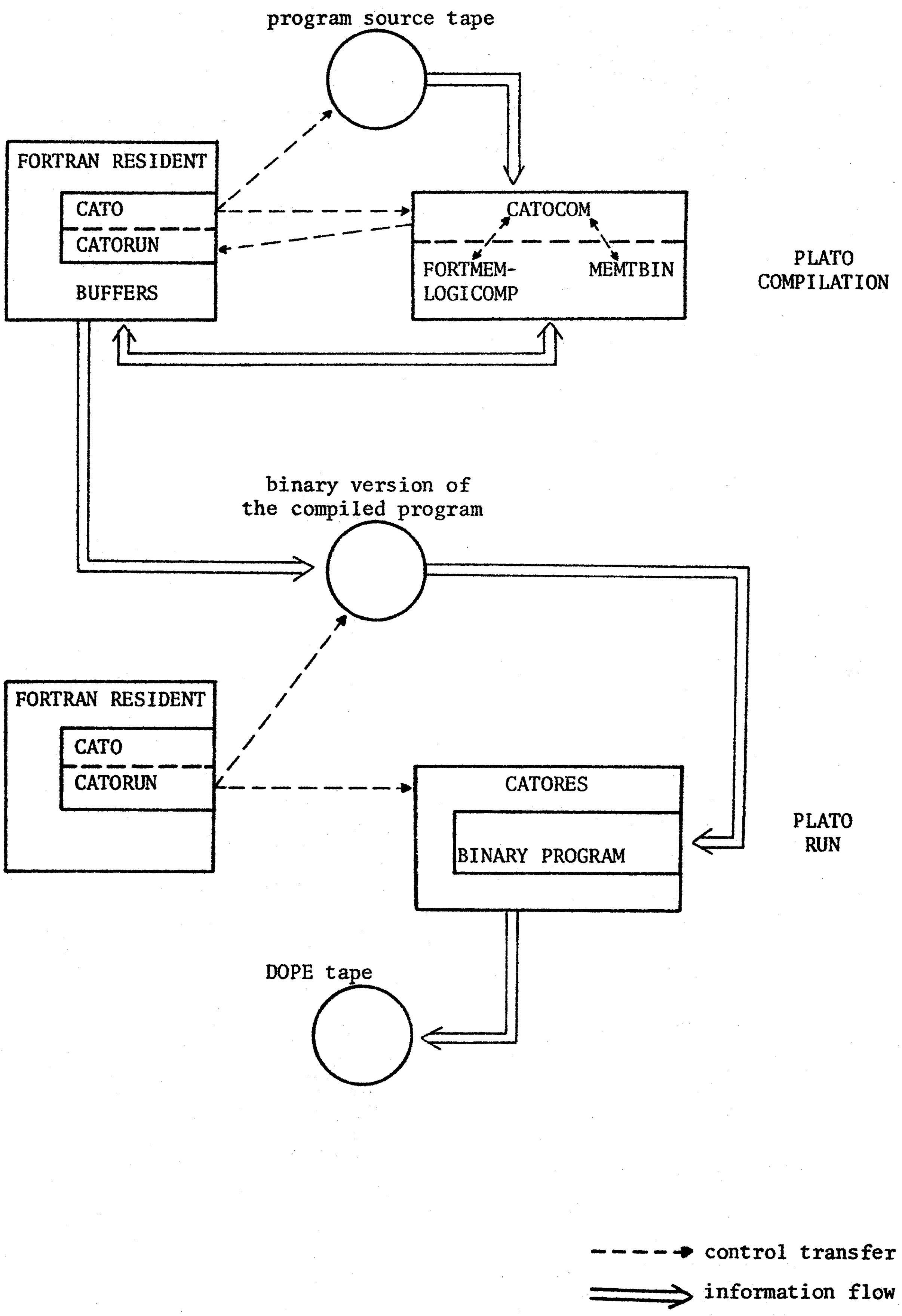
3. MODEBANK 32 words. One word for each student. Indicates the present mode of each student. MODEBANK is initially all Ø.

4. LINKLIST 513 words. Sixteen 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. Sixteen 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.

To find the beginning address of CATORES tables consult the following list:

Loc.	Tables
4140	Beginning address of CALCLIST
4141	Beginning address of MODENTRY (the entry calc list)
4142	Beginning address of KEYMODES (the modeswitch)
4143	Beginning address of CODTABLE
4144	Beginning address of PLOTABLE
4145	Beginning address of POINTS
4146	Beginning address of GRPLIST
4147	Beginning address of GRPTBLS
4150	Beginning address of STORLIST (the list of beginning addresses for each student in student bank block)
4151	No. of words in each KEYMODES
4152 U.A.	Beginning address of STORAGE (the student bank block)
4152 L.A.	Beginning address of COMMON


program source tape
FORTRAN RESIDENT
CATO
CATORUN
BUFFERS
CATOCOM
FORTMEM-LOGICOMP
MEMTBIN
PLATO COMPILATION
binary version of the compiled program
FORTRAN RESIDENT
CATO
CATORUN
CATORES
BINARY PROGRAM
PLATO RUN
DOPE tape
control transfer
information flow

PLATO MASTER TAPE LISTING

```
resident, 0, 3736,0
catores , 0, 6207,0
catocom , 7,16245,0
fortmem , 4,11507,0
logicomp, 0, 3054,0
connect , 2,   22,0
transfr ,12,  166,0
inputcdi, 2,  600,0
outputcd, 2,  600,0
inputbin, 2,   26,0
outputbn, 2,   27,0
floatfix, 0,   13,0
fixfloat, 0,   25,0
pause   , 0,   32,0
switch  , 6,   26,0
stop    , 0,    6,0
errorext, 0,   36,0
efmark  , 1,    7,0
rewindmt, 1,    7,0
backspw , 1,    7,0
eflofflo, 2,  177,0
efloffix, 2,  245,0
efixffix, 2,   60,0
sinf    , 1,  107,0
asinf   , 1,  143,0
cosf    , 1,  113,0
acosf   , 1,  150,0
tanf    , 1,  130,0
atanf   , 1,  101,0
ctnf    , 1,  146,0
actnf   , 1,  100,0
secf    , 1,  114,0
asecf   , 1,  141,0
cscf    , 1,  117,0
acscf   , 1,  134,0
sinh    , 1,  113,0
asinh   , 1,  144,0
cosh    , 1,  102,0
acosh   , 1,  163,0
tanh    , 1,   76,0
atanh   , 1,  101,0
ctnh    , 1,  105,0
actnh   , 1,   77,0
sech    , 1,   76,0
asech   , 1,  200,0
csch    , 1,  115,0
acsch   , 1,  143,0
sqrtf   , 1,   72,0
expf    , 1,   76,0
logf    , 1,  103,0
log10f  , 1,  111,0
```

```
floatf  , 1,   14,0
xfixf   , 1,   25,0
absf    , 1,    5,0
xabsf   , 1,    5,0
intf    , 1,   23,0
xintf   , 1,   25,0
signf   , 2,   11,0
xsignf  , 2,   11,0
modf    , 2,   27,0
xmodf   , 2,   10,0
dimf    , 2,    7,0
xdimf   , 2,    7,0
step    , 1,    7,0
xlocf   , 1,    4,0
transmod, 4,  127,0
whip    , 2,   30,0
chain   , 2,    5,0
start   , 0,    2,0
exit    , 0,    2,0
time    , 2,   70,0
clock   , 0,   11,0
limit   , 2,   25,0
alarm   , 1,   30,0
trouble , 4,  751,0
bindump , 3, 1010,0
list    , 2,   70,0
locate  , 1,   64,0
diffeq  , 6,   63,0
dump    , 4, 1165,0
reperf  , 0,   54,0
deq     ,11, 2210,0
matinv  , 5,  700,0
fortbin , 6,14000,0
maptbin , 3, 1200,0
fortmap , 4,12200,0
mapfmp  , 0, 4607,0
mapcdp  , 0, 4546,0
copys   , 3,   32,0
copy    , 2,   24,0
verify  , 3,   45,0
transfer, 3,   47,0
vfylib  , 3,   72,0
copyp   , 1,   21,0
edit    , 3,  336,0
getput  , 4,   50,0
set6    , 1,    6,0
stbnk   , 3,   14,0
rdbnk   , 3,   14,0
reset6  , 0,    4,0
```