ANIMATION & SPECIAL EFFECTS

Copyright June 1981	Cinetron Computer Systems, Inc.
Revised June 1982	6700 I-85 North, Bldg 1B
	Norcross, Georgia 30093-1096
Original Owner:	Kinetics, 401 West Ontario Street – Chicago IL 60610
Revised ©May 2015	Charles Vaughn

Cinetron 1100B Manual

Table Of Contents

Contents

Cinetron 1100B Manual

Table Of Contents

Table Of Contents

SYSTEM OVERVIEW

Section 1-1

All advanced capability Cinetron Systems work in much the same way. The principal difference between systems is found in the range of commands they contain and the numbers of axes they control rather than the way in which they function. Appendix "A" of this manual contains a list of Series 1100B system commands. Commands which are common to other systems (800B, 900B) are identified as such. All the commands in this manual are valid in Cinetron 1100B systems equipped with Series IV Streak/Scan software.

The system is responsive to commands given to it by way of an input terminal or a PC (Personal Computer running the Cinetron terminal software, “Miracle”) and/or from various push-button switches. A Terminal is drfined here as any communications device capable of establishing a two way data stream with the Cinetron main system. For the remainder of this manual, references to “Terminal” are equivalent to any means of communicating with the main Cinetron system via a keyboard or ASCII data stream; this could be a terminal device of any type of PC running software that emulates the function of an input terminal.

When the system is “looking at" or reading the Terminal, it issues a blank space-over and reads any data sent from the Terminal keyboard. This blank space sent by the system is a prompt indicating that the system is anticipating a typed command from the user. If a PC is used as the input device for the system, the PC system software will issue a “Happy Face” (☺) icon on its display rather than a space.

If the system does not respond when you strike a key on the Terminal then it is either running motors or shooting the camera or it is "Waiting", and looking at the various system switches. When the system is reading the Terminal it is not looking at the switches and visa-versa. When the system is running motors or shooting, only 2 switches are monitored: "EMERGENCY STOP and REVISE", and the Terminal is not monitored.

States of System Operation

1. Data Input ---------- Reads the Terminal device for data to be typed in

2. Waiting ------------- Reads manual push button switches

3. Running ------------ Reads "Revise" and "Emergency Stop"

4. Error Suspended - Reads "Continue" and "Revise"

The Error Suspended state occurs when the system detects an error. When an error occurs, the system notifies the operator with a message via the Terminal. The message is dependent upon the error, and the system suspends operation until the operator signals via the "Continue" button that it is alright to continue or the "Revise" or "Emergency Stop" button to indicate that it is not.

The user should also refer to the section in this manual referring to the "VERIFY" switch.

Extensive error handling, trapping and identification routines are included in the system to assist the operator in running the system.

A "Help" routine is included to explain errors briefly and to reference sections of this manual for detailed explanation.

Data are inspected when entered and whenever possible entry errors are found and identified in a concise, friendly manner.

SYSTEM OVERVIEW

Section 1-2

The system controls each axis of motion via a precision stepping motor. A stepping motor allows the system to position a motor to 200 or 400 discreet positions per revolution. The system may also be supplied with Servo motors in some cases. The positional aspects of a Servo may be considered by the operator as equivalent to using stepping motors, however, the number of available positions per revolution may differ. Either method of control allows the system to keep track of the exact position of each axis. The system also controls the camera shoot function, and optionally, camera forward-reverse and high speed, as well as capping shutter, automatic platen lift, color wheels, star filters and other effects devices.

Every effort has been made to make the Cinetron the most capable and, at the same time, the simplest possible effects and animation control system.

Try to think of your system as an experienced and eager assistant, ready and willing to do your every command. Remember, however, regardless of how "smart" the system may appear, **it's only a machine, and as such, is capable of doing only exactly what you instruct it to do**.

Try to have confidence that your Cinetron will do only exactly what you tell it.

It is recommended that you read through this manual twice. The first time through, don't worry if you don't understand the subtlety of each command. The next time though, try to understand as much as possible how each instruction operates. Use the manual's Index and Appendices as well as the cross references. **DO NOT TRY TO LEARN HOW AN INSTRUCTION OPERATES BY SHOOTING A DEADLINE JOB**. That's like testing the safety on a gun while pointing it at your head! Give yourself a chance, give the system a chance, and both will be happier.

Although every effort has been made to comprehensively cover the operation of this system, there are many subtleties involved in its operation, and in the type of effects it can create. Cinetron provides "hands on" training for system operators and you are encouraged to take advantage of this training to get the most out of your investment.

DATA FORMAT CONVENTIONS

Section 1-3

Within this manual several conventions are followed.

In the following text, the term “System Command” and “Identifier” are virtually synonymous. The identifier is the 2 or 4 letter initial portion of a command that identifies what you are asking the system to do. For example, in the command that asks the system to run, the letters “RU” are the identifier portion of the command: **RUn**.

Where a data format is given, the capitalized letters of the data identifier indicate the mandatory portion of the identifier. Normally on the first 2 letters on the command line are required. However, you should note that there are several commands which require 4 letters. The lower case letters indicate optional characters and are included in the manual for the sake of clarity. There is no need to use upper and lower case when operating the system as the system will generally disregard non-essential portions of the command.

Data formats include a command (data) identifier and (depending upon the command) one or a series of parameters separated from the identifier and each other. Typically, the data separator will be a comma. Where parameters are optional, they are shown enclosed by brackets.

<u>In any data command which has parameters (required or optional) the literal identifier must not exceed 5 characters or an error will occur</u>.

Data parameters consist of unsigned integer values (no decimal points or negative numbers). The values may range from 0 to 32767.

It will be noted that the largest number (32767) also contains 5 digits.

The fact that 5 letters (for the identifier) and 5 digits (for the parameters) are the maximum, allows the system to assume that if an entry allowing parameters is being entered and if more than 5 consecutive numbers or digits are found without a separator (. , or /), then some kind of error exists. The system will respond: **DATA SEPARATOR MISSING**

Please note that the alphabetical character 'O' is not the same as the digit '0' (zero), and that the lower case 'l' is not the same as the digit '1' (one). These are common errors made by some new operators who type by the touch system. If the system is expecting a digit and sees an alphabetical character, it will respond: **ILLEGAL CHARACTER**

In either case discussed, the system will space over to the error with periods (.) and point to the offending character (^).

EXAMPLES:

1. **CUrr, 10J**
   ....^ **ILLEGAL CHARACTER**
   an error because J is not a digit (10J is not a number)

2. **CUrr100**
   ....^ **DATA SEPARATOR MISSING**
   an error because the identifier CUrr is not separated from the parameter (100) by a . , or / this caused more than 5 consecutive characters to be read without a separator.

3. **CU/1**
   O.K. no error

4. **cu, 1**
   IGNORED #1 (Response)
   illegal because the ***identifier portion must be upper case characters****.*

When a line of data is typed to the system, it is not examined by the system until the operator strikes the "Return" or "Enter" key on the Terminal. This allows a line to be deleted or corrected if you find an error before hitting "Return" (or "Enter") key.

At any time before striking "Enter", if the "Delete" (DEL or Rubout on some Terminals) key is struck, the system immediately responds with a pound sign (#) and feeds the Terminal to the next line. None of the data on that line are entered into the system.

If more or less than the number of required parameters is entered, the system will respond: **NOT ENOUGH DATA!** or **TOO MUCH DATA!,** and the entry is ignored.

In order for the system to run, a CUrrent frame, a SHoot target frame, and a data entry terminator must be entered. (Some terminators imply the CUrrent and SHoot target frames (See AGain, REverse, ROtate for examples).

Only one command may be typed on a line.

A command consists of an identifier and required parameters.

WAIT

Section 1-4

Format: **WAit**

This identifier places the system in the waiting state. The system then "looks" at **all** manual switches, with the Hand Controller, Manual Indexer, Manual Fade/Dissolve, etc. becoming active. The system remains in this state until "Continue", "Revise" or "Emergency Stop" is pressed. Any of these will return the system to reading data from the Terminal. However, pressing "Continue" is the normal method to return to the Terminal after WAit is entered.

NOTE that Continue is also used to allow the system to **continue shooting** when suspended by a STop frame or a complex multiple pass operation such as a **HOokup** or sequence operation.

EXAMPLE: We are entering data via the Terminal and wish to gain control of the hand controller to reposition, and then return to finish typing.

ZERO	
CU,1	*Various Commands*
EAST,100,2,23,3,BO	
WAIT	*Tell the system to Wait*

At this point, the Hand Controller and Manual Switches are active, - Push "Continue" when finished repositioning and the system will continue reading input because no terminator has been entered.

SOUTH,100,2,24,5,IN	*Now we can type again*

DISPLAY CURRENT FRAME

(See General Information, CUrr)

Section 1-5

Format: **FRame**

The FRame command is a request for the system to display its CUrrent frame number. This is the number which the system considers to be the last frame exposed (that is, the frame number "showing" on the camera counter).

EXAMPLE:

FR	*Request frame*
Current frame "xxx"	*System Responds with the current frame number*
CU,123	*Set Current frame to 123*
……..	*Other commands*
FR	*Request frame*
Current frame 123	*System Responds with the number you set*

HELP

Section 1-6

Format: **HElp {,error number}**
or
?? {,error number}

When the system detects an error, it assigns a number which corresponds to that specific type of error. For example, error #1 indicates that the system could not find the first 2 letters in the list of allowed command identifiers. The error number is saved by the system until another error occurs or until you request HElp.

If HElp (or??) is typed without an error number, the system will print an explanation of the last error that the system detected. Once the error explanation is printed the system sets a "No Error" condition. Thus, if you type HElp (or??), the system will print the explanation. If you immediately type HElp (or??) again, the system will print "NO ERROR EXISTS".

HElp allows a number to be entered so that if you are looking over the hardcopy of an old job, and cannot remember what an error number on the printout means, you may use HElp to print an explanation of an error which occurred at some time in the past. At the end of explanation line, the HElp routine prints the section and paragraph numbers which refer to sections and paragraphs in this manual where a more detailed explanation can be found.

EXAMPLE:

JK	*Your entry.*
Ignored #1	*System responds - #1 indicates the error number*

HElp *You request help.*

First two letters of command illegal "xx-yy"
System response says look at section "xx" topic "yy" for more detail if needed. "JK" was not a legal command.

EXAMPLE: *You saw error #7 printed on an old printout of a job and can't remember what it is.*

HElp,7

Scan speed must be > 0. "xx-yy" *"xx-yy" system response tells you that an attempt was made to set the speed of the scan routines to 0 and the speed must be above 0. It also refers you to section "xx" topic "yy" for a detailed explanation.*

WRITING COMMENTS OR "NOTES"

Asterisk

Section 1-7

Format: *** as first letter on a line**

If you enter an asterisk (*) as the first letter on a line, the system considers that what you are writing is just a comment or note and does not attempt to decode the remainder of the line. Comments are sometimes helpful when a hard copy or disk file copy is being made of the operator's entries. Comments are also used in disk files of commands that are created offline or in system transfer files to convey instructions to an operator.

NOTE: SEE ALSO VERIFY

EXAMPLE:

CU,1	*System*
SH,24	*Commands*
***This is pass 1 Red Filter**	*Comment no system action*
GR	*System command*
SH,50	*System command*
***This is pass 2 Change to Green Filter**	*Comment*
GO	*System command*

AXIS-MOVEMENT CONVENTIONS

Section 1-8

The 800B, 900B or 1100B system is capable of storing up to eight (8) commands for each of 15 axes (8 axes for 900B or 800B). These commands are stored in their respective motion buffers (storage areas) in the same sequence as you enter them. The sequence in which you write commands is not *usually* important. **The system will access or not access a movement based on the start and stop move frames you have entered, and on the CUrrent frame the system has at any given time.**

When ZEro is typed, all the axis-motion data are copied to a backup buffer (See RESTore), and the motion buffer is cleared to 0 with each axis set to enter data at the first position in its respective buffer area.

It is important to note that the 8 commands refer to a particular axis, not a particular direction. That is, UP and DOWN are different directions which both refer to and are stored in the same axis (zoom in this case) buffer. Therefore, 6 up motions and 2 down motions would use all 8 positions for the zoom axis.

After the eighth command to a given axis is sent to the computer, two things will happen: First, the system will notify the operator that the buffer is full by typing "**MOTOR BUFFER FILLED**"; next, the system will reset to write the next command entered over the first command given to this axis replacing the first command. This allows data to be entered indefinitely without ZEroing. So long as the frame counts used don't overlap.

EXAMPLE:

After ZEro is typed, 10 commands are given to the Zoom axis (without ZEro being typed again, of course)

The system motor buffer will look like this:

Zoom Buffer Command Line #	Contents
1st	*9th Command Given*
2nd	*10th Command Given*
3rd	*3rd Command Given*
4th	*4th Command Given*
5th	*5th Command Given*
6th	*6th Command Given*
7th	*7th Command Given*
8th	*8th Command Given*

If another command is typed, it will be written into buffer command line 3 and so forth.

AXIS MOTION COMMANDS

(See the list of Axis Motion Commands in Appendix "B" for the identifiers for each axis)

Section 1-9

As with most system commands, an axis command may be typed in any order and at any convenient time. Whether or not the system will execute the movement depends upon whether the system CUrrent frame is between the start move frame and stop move frame as entered for a given axis.

The axis identifier is the 2 letter (minimum) code which identifies the axis and its direction. For example, UP and DOwn both identify the Zoom axis.

The distance refers to the **relative** distance to move the axis. This distance is a function of the scale of the system (see VEeder and DScale), and normally represents hundredths of an inch. (An entry of 100 = 1 inch.)

The start move frame is the CUrrent frame number where the motion is to start.

The stop move frame is the CUrrent frame number where the movement is to stop.

Ease in frames are the number of frames you wish the system to use to taper or ease the movement up to speed (accelerate).

Ease out frames are the number of frames you wish the system to use to decelerate the motion to a stop.

NOTE: As a convenience, and to maintain format compatibility with earlier systems, the ease in and ease out frame entries for inverse sine tapers may be replaced by the following:

# of frames , In
# of frames ,   Out
# of frames , Both

Where ***10, In*** is equivalent to 10, 0 meaning to ease in 10 frames with no ease out.

10, Out is equivalent to 0, 10 meaning to ease out 10 frames with no ease in 10.

10, Both is equivalent to 10, 10 meaning to ease in and out in 10 frames.

Note that in the cases above, an ease out of 0 is equivalent to an ease out of 1

NOTE: See logarithmic and Exponential move section for other optional ease entries.

There are several logical conditions that must be met for the movement described to make sense when the system tries to execute it.

1) The start move frame number must be smaller than the stop move frame

2) The sum of the ease in and ease out frames may not be greater than the difference between the start move and stop move frames.

AXIS MOTION EXAMPLES:

1) **EZ,100,2,12,1,1**
Error - EZ is not a valid identifier

2) **UP,1000,1,100,1,5**
OK – means to move up 1000 units on the zoom counter - start moving when the CUrrent frame is 1 and stop when it reaches 100. Ease in 1 frame (the same as no ease in) and ease out 5 frames.

3) **UP,1000,1,100,5,Out**
OK - exactly the same as example 2

4) **UP,1000,1,100,50,50**
Error - The sum of the ease in and ease out (50+50=100) is greater than the difference between the start and stop frames (100-1=99)

5) **UP,1000,100,1,5,6**
Error - The start frame is larger than the stop frame

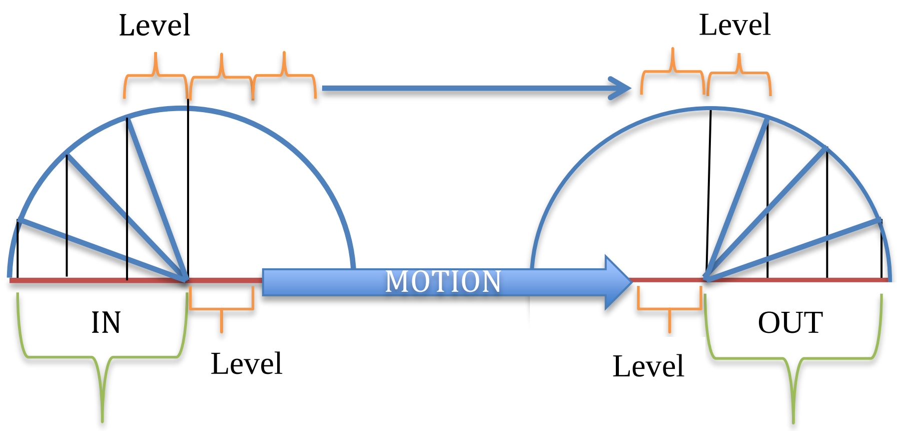
NORMAL EASE TAPERS (INVERSE SINE)

(See also Logarithmic Tapers, Exponential Tapers)

Section 1-10

The default ease in and out tapers used by the system are based on an inverse sine relationship. Experience has indicated that this type of taper yields the smoothest, most life like acceleration and deceleration of most motions.

It might be helpful to note that the Inverse Sine Tapers may be represented by the following diagram:



In this illustration, a four frame taper in and out are shown along the red line. The last frame of the taper in and the first frame of the taper out are equal to each other and to the distance moved in every frame of the attained (level) speed portion of the move between them. Technically, only three frames of acceleration and deceleration are created since the fourth equals the “steady rate” or “level move” of motion between them. It is permissible to create a move in which all frames are tapering for example: UP,200,1,11,5,5 all 10 frames are tapering (actually only 8 are tapering since the last taper in is equal to the first taper out).

Overlapping the frame counts on 2 or more commands to an axis will cause the system to compute the composite sum or difference of the commands.

EXAMPLE:

1) **UP,1000,1,100,5,5**
   DOWN,1000,1,100,5,5

   Causes the zoom not to move since the moves are identical but opposite in direction.

2) **UP,1000,1,100,5,5**
   DOWN,1500,1,100,5,5

   The result is a 500 unit down move

3) **UP,1000,1,100,5,5**
   UP,500,50,100,5,5

   Causes the zoom to ease up from current frames 1 to 6, maintain a level speed to frame 50 and then accelerate from frames 50 to 55 attaining a speed twice as fast, then to slow down from 95 to 100 and stop. The total distance traveled is 1500.

4) **NORTH,1000,1,100,5,15**
   EAST,100,1,100,20,15

   Causes the North to accelerate faster than the east motion, causing a north curve to be generated with both axes decelerating diagonally.

5) **NORTH,1000,1,100,5,15**
   EAST,100,24,120,10,24

   The North will move for 23 frames before the East starts, and finish 20 before the East finishes. This produces a straight line North movement, gradually changing to an arching North/East motion, the changing to a diagonal East motion, which changes to an arching East motion until stopping on frame 120.

LOGARITHMIC TAPERS

Section 1-11

A continuous logarithmic taper IN can be achieved by setting the taper in length to 0 and typing LOG as the description.

EXAMPLES:

Sine -- **EAST<100,2,10,6,IN**

LOG -- **EAST,100,2,10,0,LOG**

Each successive frame accelerates as the log^e of the previous distance output creating a motion that has a logarithmic speed increase.
There is no Logarithmic taper out available; if however, a scene is photographed with the film and axis motion directions reversed, the same visual effect will be created.

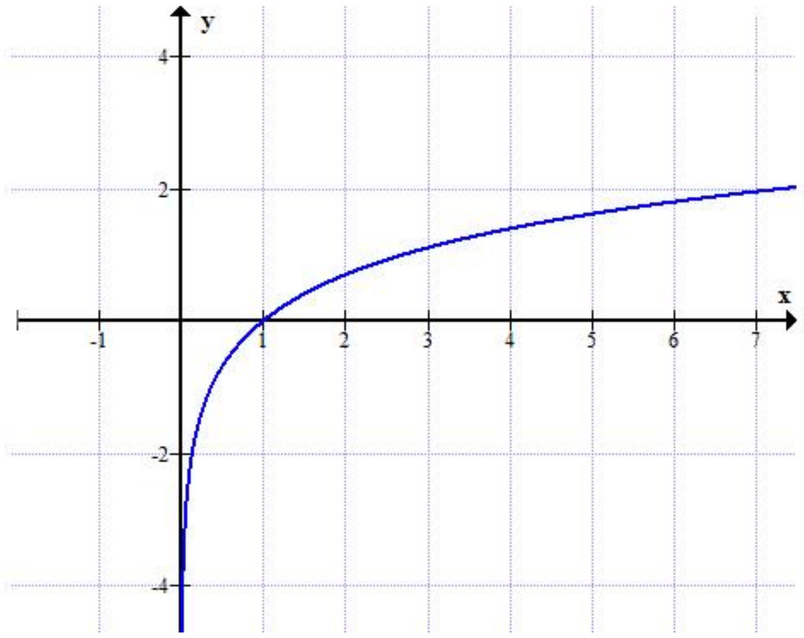


Illustration of a Logarithmic Function (x=time, y=speed)

EXPONENTIAL ZOOMS

Section 1-12

Format: **EZoom, Zoom #, Zoom Top Position, Zoom Bottom Position**

The zoom number entered is the Motor Data Buffer line number on which a conventional zoom **has been entered**. The EZ command will cause this zoom to be treated as exponential in nature. See Sections 1-8, 1-9 and 1-23 for information about Motor Buffer lines.

The top position is the zoom counter reading at the highest position of the zoom, the bottom is the zoom counter reading at the lowest point. **It is important that the distance entered in the conventional zoom being converted to exponential correspond to the difference between the high and low points in that original move**.

If the conventional zoom is tapered, the exponentiation of it will also be tapered.

Once an exponential zoom has run it must be ZEroed or removed with OM,X or EZ,0,0.0. It can then be re-run by re-entering the EZ command to re-run as an exponential or without EZ to run as a conventional zoom.

All exponential off center moves will be matched to the image size and position of the EZ being run when the frame counts of the off center move are reached.

Exponential Zooms are calculated so that zooms which start and end at different heights appear to move at the same rate provided the number of fields moved and the number of frames used are equal or are in proportion.

EXAMPLE: We wish to make a 20 frame zoom on 2 pieces of artwork. The artwork is prepared so that the first zoom is from 14 fields to 8 fields, the second from 12 fields to 6 fields. Each zoom covers 6 visual fields of motion.

14 fields = Counter 6000	
8 fields = Counter 3428	*Difference=2572*
12 fields = Counter 5142	
6 fields = Counter 2703	*Difference=2439*
ZEro	*Clear everything (not always required)*
CU, 1	
DO,2572,1,11,1,1	*Move to be made exponential*
EZ,1,6000,3428	*Exponential conversion*
GO,11	*Shoot exponential move #1*
Zero	*Clear everything*
CU,1	
DO,2439,1,11,1,1	*Second 6 field move*
EZ,1,5142,2571	*Exponential conversion*
GO,11	*Shoot 2nd Zoom*

Both zooms will appear to be at the same speed because they cover the same number of frames (10) and the same number of fields (6). Therefore (if the artwork matches), the 20 frames will appear as 1 continuous zoom. Note that if either zoom is tapered the other must be correspondingly tapered at the opposite end to match speed at the center. That is, if the first is tapered in 5 frames, the second must be tapered out 5 frames. Variations in manually cut focus cams can affect the accuracy of these zooms. Animation stands with Cinetron Computer Follow-focus units will match exactly.

Exponential zooms have the characteristic of moving at an apparent constant rate as the camera height changes. This will eliminate the apparent speed increase that is observed as the camera zooms in and the decrease when zoomed out. This is a major use of this type of taper.

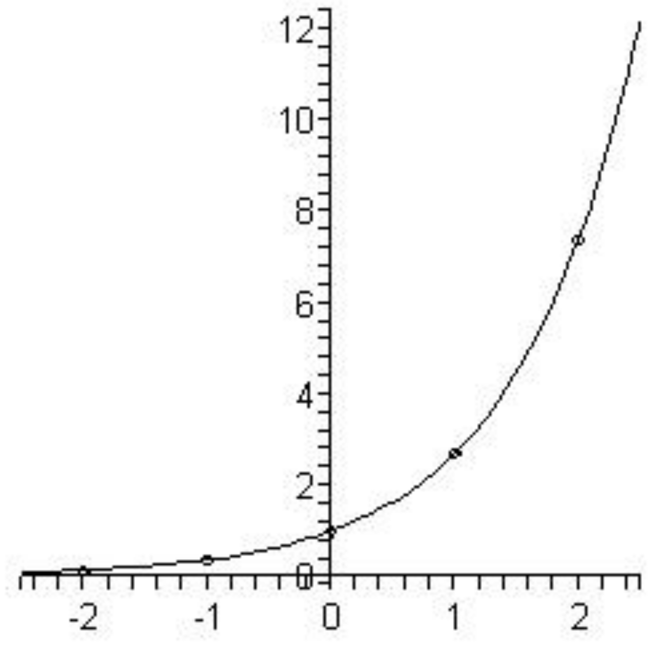


Illustration of an Exponential Function (time along EW axis Speed on NS)

EXPONENTIAL OFF CENTER

Section 1-13

The other movements to create a compound exponential move are entered in the format: **AXIS IDentifier (NOrth,SOuth,etc),Distance, Start Frame, Stop Frame, 0, EXp**

Exponential off center movements will match apparent speeds with respect to field size and the numbers of frames, as do the exponential zooms, providing that the off-center motion <u>in fields</u> is equal.

FADE/DISSOLVE

Section 1-14

General Information

Generally, the Fade/dissolve is treated like other motor axis commands with Fade and Dissolve comprising one axis. Like the Axis Motion Commands, the Fade/Dissolve Commands are stored in a memory buffer. This allows you to enter more than one fade or dissolve command at a time. Unlike the other motion axis buffers, the Fade/Dissolve allows only 7 commands (as opposed to 8 for a normal axis) to be entered before the buffer is filled. When the 7th command is typed, the system will respond: FADE/DISSOLVE BUFFER FILLED, and the next Fade or Dissolve entered will be written over the first one entered.

There exists an 8th Fade/Dissolve position in the buffer, but this position is reserved for the manual Fade/Dissolve unit (if installed) and is accessible only from the manual Fade/Dissolve switches.

When the system reaches the CUrrent Frame where a Fade or Dissolve is indicated to start, it inspects the current position of the shutter to see if it is in the correct position to execute the indicated command. For example, to fade out the shutter must be open, to fade in it must be closed.

If the shutter is in the wrong logical position to properly execute the command (for example open for a fade in), then the system will respond: **INVALID FADE DISSOLVE**

The command line which was illegal is erased, and the system will wait until you push "Continue" before proceeding.

You can recover this situation by entering the correct Fade/Dissolve on the manual Fade/Dissolve unit before pushing "Continue". If you wish to **REVISE** rather than continuing, push REVISE **& hold the switch down while pressing "Continue"**.

When the system runs a Fade or Dissolve, it erases the command as soon as it is executed. **Therefore, you cannot "AGAIN" or "REVERSE" a fade or dissolve**.

EXCEPTIONS:

1) If **AUdition** is used, Fades and Dissolves are **not erased.**

2) If **MUlti, SMulti** or **CYcle** are used, Fades and Dissolves are **not erased.**

3) **MANUAL FADES AND DISSOLVES are always erased**.

FADE OR DISSOLVE

Section 1-15

Format: **FAde , frame to start, length, IN (or OUT)**
DIssolve ,frame to start, length, IN (or OUT)

The length of a fade can be any number above 2 frames to a practical maximum of 2000; a dissolve 3 frames to 2000.

EXAMPLE: To fade at frame 233 for 16 frames fading out

FA,233,16,Out

FADE COMPENSATION

(See Also Dissolve Compensation)

Section 1-16

Format: **FComp, open end jump, closed end jump**

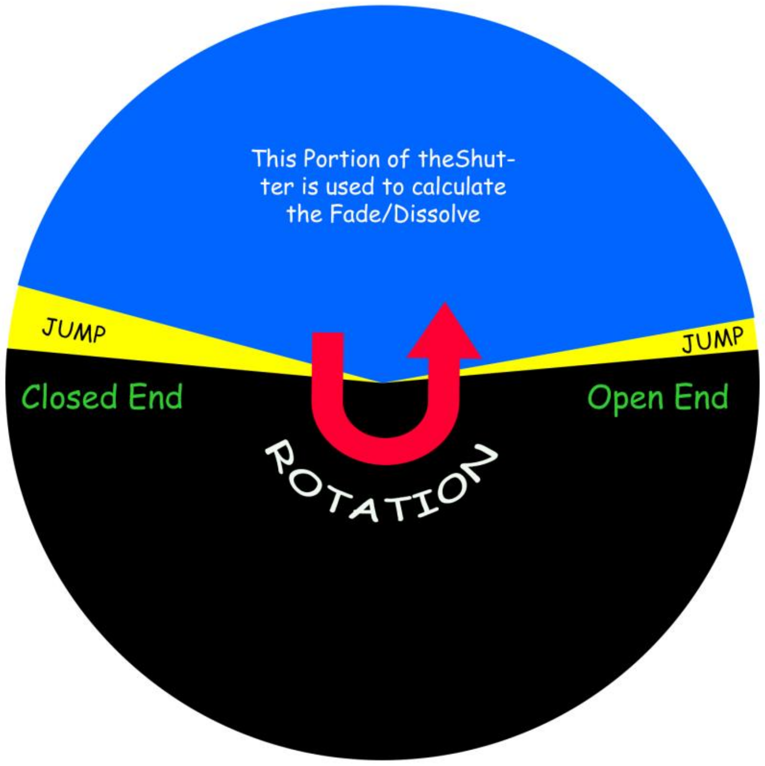
General Information

To allow for differences from one scene to another and for differences in film stock and processing, the Fade Compensation commands may be used,

The Fade Compensation Command is used by the system only to execute fades and has no effect on dissolves.

The values for **Fade Compensation** are **not affected by ZEro**.

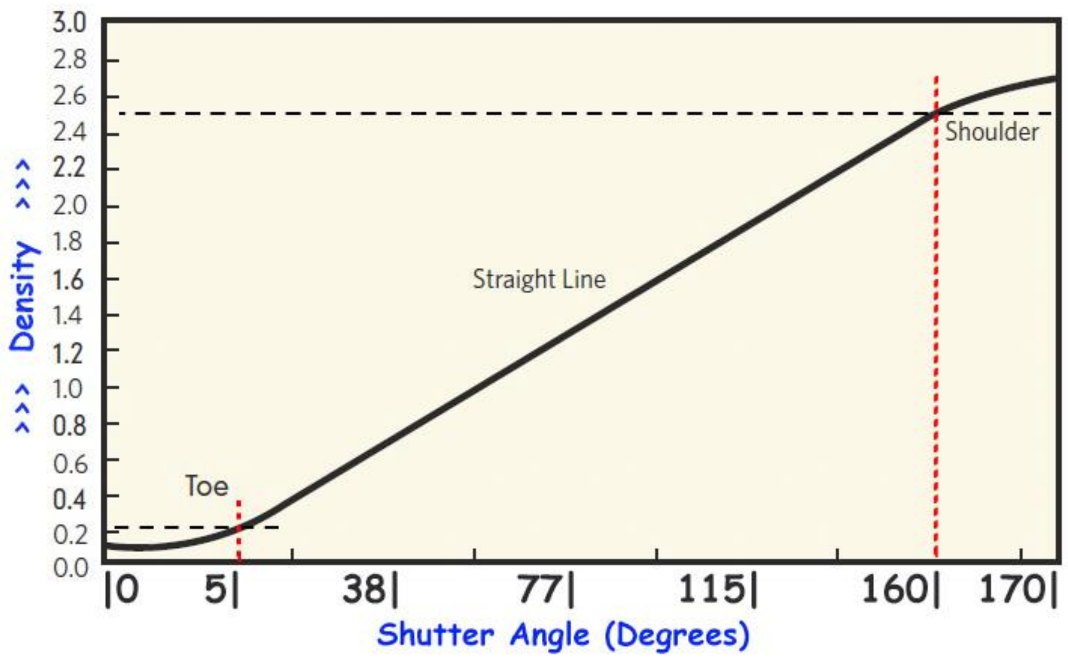
The Fade Compensation Command reshapes the fade curve by compressing the amount of the shutter that is used for the fade. This is accomplished by forcing the shutter to take a predetermined "Jump" on one or both ends of the fade, and then spreading the fade calculations over the remaining shutter angle.



In the diagram we see a representation of a 170º shutter. The open end jump is 5º of the shutter and the closed end jump 10º of the shutter. Therefore, 155º of the shutter is actually used to calculate the fade. When a Fade **in** is executed, on the first frame of the fade, the shutter jumps 5º plus the amount calculated for the first frame of the fade. The fade then continues normally until the last frame is calculated, then the shutter “jumps” opens fully on the last frame.

The main purpose of the Fade Compensation is to shift the fade into the "straight" line section of the film stock characteristic curve, using the open and close end jumps to avoid the "shoulder" and "toe" regions and to nullify any ‘slack’ in the gears used to vary the shutter angle.

The following diagram represents the characteristic curve of an imaginary film stock, with density plotted against shutter angle - you will see that the first 5º of opening and last 10º of opening have little effect on the final density. Therefore, if frames were exposed here, little, if any, fading would be observed.



The ideal Fade Compensation for this film stock on a "perfect/average" scene would be a 10º closed end jump and a 5º open end jump if no gear slack is present. NOTE: The designations, Open End and Closed End are derived from whether or not the shutter is just opening or just closing, the “Jump” is manipulated by the system depending upon whether the fade or dissolve is actually at the most closed or most open portion of the effect and the direction of the effect (IN or OUT).

FORMAT: **FComp, open end jump, closed end jump**

The actual values for the entries for closed and open end jumps are derived from a separate print-out supplied with your system. This print-out was specifically tailored to the shutter in your system, and gives a number to enter which corresponds to a number of degrees. **You may want to increase the settings to compensate for film sensitivity characteristics**.

Fade Compensation is changed by re-entering a different compensation and removed by entering: **FC,0,0**.

DISSOLVE COMPENSATION

(See Fade Compensation)

Section 1-17

Format: **DComp, Closed End Jump, Open End Jump**

Dissolve Compensation is exactly like fade compensation with one important difference. Since in a dissolve, the combination of Dissolve In and Dissolve Out must always equal 100% exposure, **the numbers entered for open and closed end jumps are normally the same**.

You may encounter occasional situations where the open and closed ends are not the same for the screen result desired. These are primarily situations where there is a great difference in the brightness of the two scenes being dissolved. Experimentation will dictate these situations.

DC has **no effect on fades**

DC is **not removed by ZEro**

To remove Dissolve Compensation enter: **DC,0,0**

OVERLAPPED DISSOLVES

Section 1-18

Format: **OVer, x**
x = The amount the shutter is to use to calculate dissolves
This amount is derived from a separate print-out
supplied with your system.
(See also Fade Compensation, Dissolve)

To remove overlapping enter **OVER,0**

OVER is **not affected by ZEro**

OVER **has no effect on Fades**

When the OVer value is greater than 0, the system considers all dissolves to be overlapped dissolves.

When this condition exists, the system will attempt to execute a dissolve regardless of the current shutter position. Therefore, it is possible that the system will try to do a dissolve out with the shutter already closed, or a dissolve in when the shutter was already open. **When the overlap is used, it is the operator's responsibility to keep track of the validity of dissolves entered.**

This is because the free-form input that the system uses prevent it from being able to tell whether or not a fade or dissolve command is logically valid at the time a command is entered.

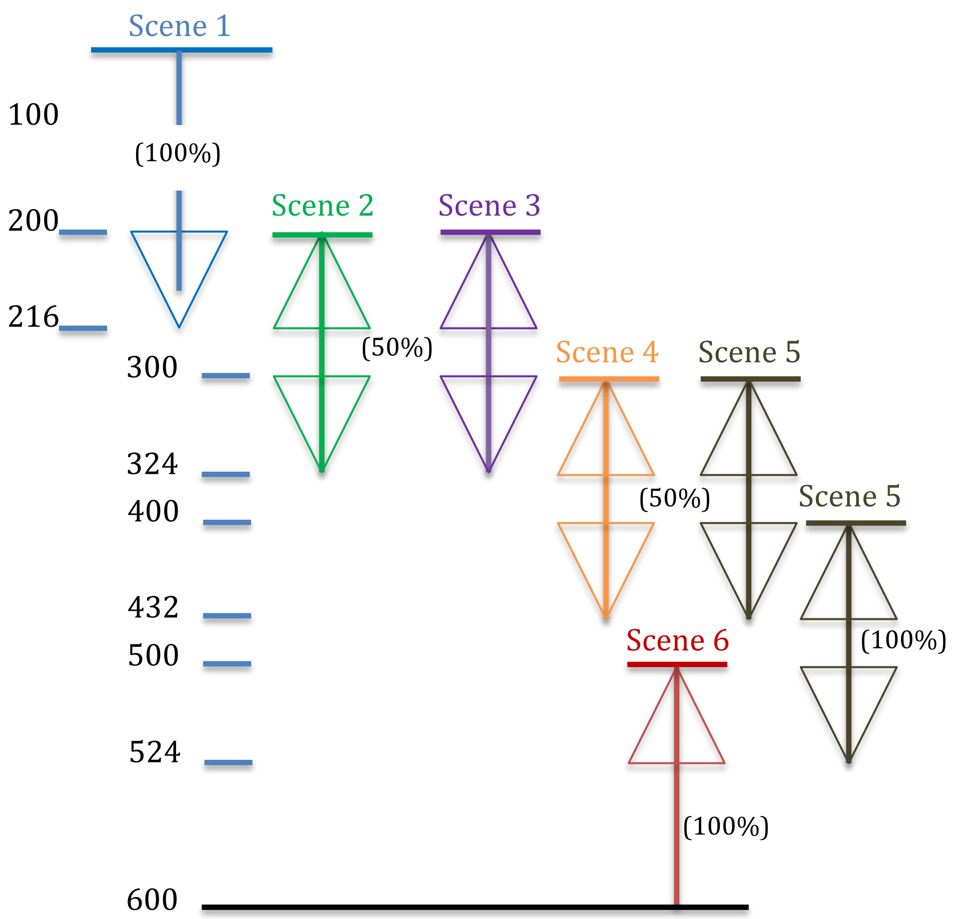
OVERLAPPED DISSOLVE EXAMPLE

For our example, the OVer values are used in degrees

OVer, 85 =85º and the full shutter opening is 170º

We wish to shoot a scene which dissolves to 2 scenes (Scenes 2 & 3), superimposed over each other. These 2 scenes dissolve to 2 other superimposed scenes (Scenes 4 & 5), which then dissolve to one of the two (Scene 5), which then does a normal dissolve to the last scene, Scene 6

LAYOUT FOR OVERLAPPED DISSOLVE EXAMPLE



From the layout we can see that Scene 1 requires 100% exposure (OVER, 0). Scenes 2 & 3 each require 50% exposure (OVER, 85 or 1/2 shutter or 85º). Scenes 4 & 5 require 50% exposure until frame 400 where Scene 5 requires 100% exposure as does Scene 6. Here are the Commands required for this example:

ZERO	- *(Not Necessary)*
Over, 0	*(Assuming maybe it was not already 0.)*
CU,100	*Establish the current frame count*
DI, 200, 16, Out	*Dissolve out Scene1*
SH, 216	*Set Target frame at End of Scene 1*
GR, 200	*Shoot to Target frame 216 & Rewind to 200*
Over, 85	*Set for 1/2 shutter*
DI, 200, 16,IN	*DIssolve in Scene 2 to 50%*
DI, 300, 24, Out	*DIssolve out Scene 2*
SH, 324	*Set Target frame at End of Scene 2*
GR, 300	*Shoot to Target frame 324 & Rewind to 300*
DI, 300, 24, In	*DIssolve in Scene 4 to 50%*
DI, 400, 32, Out	*DIssolve out Scene 4*
SH, 432	*Set Target frame at End of Scene 4*
GR, 200	*Shoot to Target frame 432 & Rewind to 200*
DI, 200, 16, In	*DIssolve in Scene 3*
DI, 300, 24, Out	*DIssolve out Scene 3*
SH, 324	*Set Target frame at End of Scene 3*
GR, 300	*Shoot to Target frame 324 & Rewind to 300*

Dissolve Commands required (continued)

DI, 300, 24, In	*DIssolve in Scene 5 to 50%*
DI, 400, 32, Out	*DIssolve out Scene, 5*
SH, 432	*Set Target frame at End of Scene 5*
GR, 400	*Shoot to Target frame 432 & Rewind to 400*
	finish Scene 5 at 100%
Over, 0	*Remove Overlap*
DI, 400, 32, In	*DIssolve Scene 5 back in to 100%*
DI, 500, 24, Out	*DIssolve Scene 5 Out*
SH, 524	*Set Target frame at End of Scene 5*
GR, 500	*Shoot to Target frame 524 & Rewind to 500*
DI, 500, 24, In	*DIssolve in Scene 6*
GO, 600	*Shoot to End*

If it is desired to shoot in a different order than used in the example, it may be necessary to cap the shutter and dissolve out to attain the proper shutter position. This is because the **OPen** command is always in degrees and is not quite as accurate as the technique used for overlapping. If you wish to insure absolute accuracy (probably not really necessary), run a dissolve with the shutter capped, to adjust the position of the shutter for an overlap.

OPEN

Section 1-19

Format: **OPen (, # of Degrees)**

The OPen command causes the Fade/Dissolve shutter to run to a fully open position.

OPen may be used to insure that the shutter is in fully open position before executing a fade or dissolve out.

OPen **<u>MUST</u>** be used prior to a Command to Strobe out (XO), CYcle, MUlti or SMult when the shutter is to start opened.

OPen may optionally contain a number of degrees to open the shutter. This feature can be used to set the shutter to a specified amount of exposure less than 100%

EXAMPLES:

OPen Runs Shutter Open	*- Open Fully*
OPen, 85 Runs Shutter to 85º	*- Open to 85º* *(50% Exposure)*
Open, 10 Runs Shutter to 10º	*Open to 10º*

BLACK

Section 1-20

Format: **BLack**

Causes the Fade/Dissolve shutter to fully close as soon as the command is received. BLack may be used to insure that the shutter is fully closed before Fading or dissolving in.

BLack **MUST** be used before a strobe in or strobe both (XI, XB), or a CYcle, or SMulti when the shutter is to start closed.

OMIT

Section 1-21

Format: **OMit, Single letter identifier for axis or command**

OMit removes (sets to 0) all data in the buffer associated with the identifier. Appendix "I" contains the list of identifiers. In the case of an AXIS, remember that all 8 commands for an AXIS are set to 0 -- that is, if North is the identifier, both North and South are removed since they are different directions but the same axis of motion.

The system will respond "**ENTRIES GONE!**" if the format is acceptable.

Note that only 1 letter is used as an identifier for the buffer to OMit, and as soon as the system finds a valid identifier, that buffer is cleared. No further reading of the entry line is done. It is not possible, therefore, to OMit more than 1 axis per OMit call.

EXAMPLE:

OMit,Z	*Omit the Up/Down zoom axis entries*
ENTRIES GONE!	*System Response*
OMit,N	*Omit all North/South Axis entries*
ENTRIES GONE!	*System Response*
OM,X	*Omit all special routines (See Appendix "I")*

FLIP

Section 1-22

Format: **FLip, single letter identifier** (see OMit)

The format for FLip is exactly like OMit, as also, are the other single letter identifiers. Appendix "I" contains the list of OMit/FLip identifiers.

The FLip command refers only to motion axes and causes **all the commands** for the identified axis to reverse directions - North becomes South - South becomes North, etc. When the system finds a valid identifier, it responds **DONE!**

See Appendix "I" for a list of the single letter identifiers for FLip.

Since FLip refers to an **axis**, the ‘X’ and ‘P’ identifiers in the list are not valid since they refer to special routines and the platen lift action.

MOTOR BUFFER MODIFICATION

Display Buffer

Section 1-23

Format: **DBuff, x**
x = Motor number from 1 to 15

The Display Buffer command allows the operator to examine the contents of motor command buffers (See sections 1-8 and 1-9).

The number entered is the number of the motor whose buffer you wish displayed. Appendix "C" contains the list of axes and their corresponding motor numbers.

When the DBuff command is received, the system will print all the axis commands for the indicated motor. The system will also print the next buffer line on which an entry will be made when a command is written to the axis.

EXAMPLE: To print the buffer for axis #1 (Zoom)

DB,1
System responds
BUFFER CONTENTS, MOTOR #1
Next entry line > x *(x= the next line data will write to)*
DIR. DIST. START STOP EASEIN EASEOUT

This will be followed by the data in each line. The data is in exactly the same format which you would use to enter information to this axis (see section 1-9).

Note that automatic moves are placed in the axis buffers in a form which is compatible with normal commands. Therefore, a move entered as an automatic could be entered as a normal move once the DB command is used to obtain the distance entry.

SET MOTOR WRITE LINE

Section 1-24

Format: **WMotor,x,y**
x = Motor number (1-15)
y = Line number (1-8)

The Write motor command is used to force the system to enter the next axis command to the indicated axis on the buffer line number entered.

See sections 1-8 and 1-9 for a discussion of the motor buffers and axis commands.

The WM command is provided to allow a method for editing and altering data in the motor command buffers. WM is used to alter or defeat the system automatically entering data on the next sequential line of the buffer

SECTION 2

CURRENT FRAME

Section 2-1

Format: **CUrr, x**
x = The value to set as the current frame

When the system runs it keeps track of the frames it photographs and maintains this count as the CUrrent frame. It is **not necessary** (or even desirable in many cases) **for this number to match the number on the camera frame counter**. It is also not necessary to make this entry unless you want the count to differ from that maintained by the system.

Note that ZEro sets the current frame count to 0. When the system runs, it shoots from the CUrrent frame to the target (SHoot) frame. If allowed to finish, the CUrrent frame will equal the last entered target frame. While running, the CUrrent frame number determines which movements are to be run.

EXAMPLE #1: *Start at 1 shoot to 25 then to 50*

ZEro	*(sets everything to 0 this is optional)*
CUrr, 1	*(current frame set to 1)*
SHoot, 25	*(target frame set to 25)*
GO	*(command terminator, causes immediate run*)

when the system returns to the Terminal the CUrrent frame will = 25 (i.e. 24 frames exposed)

SH, 50	*(set target = 50)*
GO	*(run)*

when finished current frame = 50 (49 exposed frames 1 thru 50)

EXAMPLE #2: *Same requirements as example #1*

```
ZE
CU,1
SH,25
GO
CU,25                 (unnecessary but harmless)
SH,50
GO                    (same results as example #1)
```

Note that typing **FRame** will cause the system to print its current frame number (See FRame).

From this example, you can see that the Current frame is the number **showing** on the camera's frame counter provided the camera frame counter matched (frame 1) at the outset.
There is a difference of opinion between companies, animators and operators as to whether the frame showing when the camera is first loaded is frame 0 or 1 and whether it is exposed or not.
The logic of the Cinetron system first checks the Current frame number entered or maintained by the system to see if it needs to make a move that will occur **between** the Current frame and the next exposure. If not, the system makes an exposure, adds 1 to the CUrrent frame count and continues checking.
The camera frame counter advances after an exposure is made.
If you want to make frame 0 exposed just shoot a frame and set the counter to 0.

READ THE IMPORTANT NOTE ON THE NEXT PAGE!

IMPORTANT NOTE

Section 2-2

If it is desired to run the system from CUrrent frame 0, it is usually desirable to enter a STop frame with a value above 0.

This prevents accidentally running the system if the CUrrent frame is 0. Since ZEro sets the CUrrent frame = 0 and sets the STop frame = 0, the system will stop before making an exposure (you have, after all, instructed the system to Stop when the Current frame is 0). This behavior prevents your running without remembering to enter a CUrrent frame or a STop frame after ZEroing. If this occurs (STop and CUrrent = 0), then the system will type: **"MID RUN STOP REACHED, current frame = 0".** If you really did forget to enter the CUrrent frame, press "REVISE" and enter a non-zero current frame or merely press the "CONTINUE" button if you really intended for the Current frame to be 0.

SHOOT

Section 2-3

Format: **SHoot,x**
x = The number you wish to be the last frame exposed
(relative to CUrrent frame)
(See also CUrrent - RUn - ENd - GO-System Data Terminators)

This is the system SHoot (target) frame relative to the CUrrent frame. No film will be exposed if the current frame is equal to or larger than the shoot frame.

EXAMPLE 1:

CU,1

SH,2

GO

1 frame exposed as soon as GO is received by the system

EXAMPLE 2:

CU,431

SH,390

GO

Error current>target (nothing will shoot)

EXAMPLE 3:

CU,1000

SH,1090

GO

90 frames exposed

ZERO

Section 2-3b

Format: **ZEro** (no parameters required)

See also RESTore, RScan

Typing ZEro causes all the motion, frame, special routines and routine calling identifiers to be set to 0. The Slitscan buffer is also reset to the first DAta line and blocked from running. Any data which are set to 0 are first copied to a backup area and can therefore be retrieved by using the RESTore command

Typing ZEro twice (on separate consecutive lines) will cause this backup area also to be set to 0.

ZEro has no effect on the indexers memorized positions, scan speed, fade or dissolve compensation hookup points, scaling, absolute center, safe area, transfer file, shutter position, or dissolve overlap.

NOTE!!! It is **not** a good habit to use ZEro indiscriminately as this diminishes the potential versatility of the system command structure and reduces the power of the backup function.

RESTORE

(See also REverse, RScan, and ZEro)

Section 2-4

Format: **RESTore**
(Note that 4 characters are required.)

Typing RESTore causes the data stored in the system backup buffer to be copied into the system running area. The Slitscan buffer is NOT restored. Refer to the RScan section for details on restoring data in the Slitscan buffer. The data restored are identical to the instructions in memory the last time ZEro was typed.

When the data are restored the system types:

DATA RESTORED current frame (frame #)

The CUrrent frame of the restored data is displayed and **<u>data currently in memory is overwritten and therefore lost</u>.**

EXAMPLE:

(A) **ZEro**

ZEROED! *(system response)*

(B) **ZEro**

ZEROED!

At this point, both the running memory and the backup buffer are set to all 0's

(C) **CUrr,1**

(D) **SHoot,25**

(E) **West,100,10,25,5,5**

(F) **ZEro**

ZEROED! - *(Run buffer set to all 0's and data from the running memory buffer copied to backup)*

(G) **FRame** *(Current frame request)*

Current frame 0 *(Current = 0 caused by entering ZEro)*

(H) **RESTore** *(retrieve data from backup)*

DATA RESTORED current frame 1 *(System Response)*

(I) **GO** *(terminate data Entry and Run to frame 25)*

At this point, the system will run on the restored data and shoot from frame 1 to 25 and the table will pan West beginning on frame 10 and ending on frame 25.

AGAIN

Section 2-5

Format: **AGain**

When AGain is typed, the system responds: **READY!** then when "Continue" is pressed, the system will run.

AGain is used to repeat the last set of instructions again.

When a terminator (see list of system terminators) is typed, the system copies the current frame and target (SHoot) frame to be run to a location **unaffected by ZEro and RESTore** (however movements are affected by ZEro and RESTore). These values are placed back in CUrrent and SHoot when REverse, ROtate or AGain are typed.

EXAMPLE 1:

CU,1

SH,10

GO *-The system saves the CUrrent and SHoot frames in case AGain, REverse or ROtate is typed.*

(System shoots 9 frames Current frame =10)

AGAIN

READY! *(System Response) - REstore old CUrrent and SHoot frames.*

When continue is pressed, the system will again shoot from frame 1 to 10, and will make any moves which may have been entered. When completed the system Current Frame will be 9.

EXAMPLE 2:

ZERO

CU,1

SH,10

GO

(9 frames exposed)

SHoot,11

GO

(1 frame exposed)

AGain

READY!

When continue is pressed, 1 frame will be exposed (from 10 to 11).

NOTE: The CUrrent frame in the computer will be the same number after an AGain, ROtate or REverse, as it was at the end of the original run.

REVERSE

(See also ROtate, RECOrd, RESTore, AGain)

Section 2-6

Format: **REverse**
(Note that only 2 letters are required)

Although the first 2 letters of REverse, RECOrd and RESTore are alike, the system will respond to each different command in a different way.

When REverse is typed the system response is: READY!

REverse is exactly like AGain except that the direction of the **axis motors** are reversed. Note however, the direction of the **camera motor is not reversed**.

PLEASE NOTE:

Since REverse requires only the first 2 letters for identification, a REverse will occur if RESTore was attempted but the second 2 letters (ST) were in left off or misspelled.

EXAMPLE:

RETS	- *Meant to type REST*
READY!	- *System interpreted command as REverse*

To correct this situation- Press "REVISE" and retype RESTore correctly and FLip ALL to reset the axis motor directions back to where they were (undoing the action of the misinterpreted Reverse).

ROTATE

(See also REverse, AGain)

Section 2-7

Format: **ROtate**
(Used to execute a move already run, backwards.)

ROtate, like REverse, does NOT change the direction of the camera motor. When ROtate is typed, the directions of the original motion is reversed (Up becomes Down, East becomes West, etc.), **and** the move is executed from the previous SHoot target frame to the previous CUrrent frame. Unlike REverse, this causes a taper out to become a taper in and vice-versa.

Note: If it is anticipated that ROtate will be used on a move, it is important to arrange the movement instructions so that at least 1 frame will be shot in position before moving.

EXAMPLE 1:

CU,1

EAST,100,2,10,1,6

GO,10

Shoots 9 frames moving EAST with 6 frames tapering out. When finished, current frame will be 10.

ROTATE

READY! *(System Response)*

When "Continue" is pressed, the system will shoot 9 frames moving West with a 6 frame taper in; when finished, the system current frame will be 1.

NOTE: If the direction of the film motion was manually reversed, then when the ROtate is finished, the film and compound will be in the same positions as when the original instructions were given.

EXAMPLE 2:

CU,1

EAST,100,1,10,1,6

GO,10

Same action as Example #1 except that the table will move before frame #2 is exposed. Remember setting Current =1 implies that frame 1 is already exposed.

ROtate

READY! - *(System response)*
When "Continue" is pressed, the system will run as it did in Example #1, except that it will finish without making the move that occurred before frame 2 was exposed. This is because the target frame was set to 1 by the ROtate command, and when the frame #2 is exposed causing 1 to be the next frame, the system will complete its run because frame 1 is showing and the move in the original command is implied to occur between frames 1 and 2.

Remember - In forward the mechanical camera frame-number indicates the frame already exposed, and in reverse it indicates the <u>next</u> frame <u>to be</u> exposed.

Forward Direction	Rotated Direction
MOVE	SHOOT
SHOOT	MOVE
MOVE	SHOOT
SHOOT	MOVE

STOP
Mid Run Stop Frame

Section 2-8
Format: **STop,x**
x = The system CUrrent frame on which we wish to stop.

The STop instruction is used to define the frame on which we wish to stop shooting, and "look" at the manual switches and hand controller. When the STop frame is reached, the system will type: **MID RUN STOP REACHED Current Frame XXX** and go into a state where the manual switches are active (Waiting State).

STOP is used primarily where there is only one frame within a block which requires the operator to perform some manual action; such as changing artwork, entering a manual fade, etc.

When we're ready for the system to go back and run to the SHoot target frame, push "CONTINUE" and the system will proceed.
Pressing “REVISE” instead, will return the system to the Terminal where additional commands can be entered if desired. The current and target frames do not need to be altered unless required.

The STop frame can be any frame number, whether or not it falls between the CUrrent and target frames.

MID RUN STOP EXAMPLE 1:

CU,1

STop,24

GO,50

The System will shoot from 1 to 24 then "STop".

When "Continue" is pressed, the system will SHoot on to frame 50.

MID RUN STOP EXAMPLE 2:

CU,0

STop,100

GO,50

Will shoot from 0 to 50 without stoppng.

(See CUrrent frame section regarding the use of Stop with a

CUrrent frame of 0)

SYSTEM DATA TERMINATORS

Section 2-9

When the system is reading your commands on the Terminal, a DATA TERMINATOR COMMAND or Terminator is used to tell the system that you are finished and ready to shoot. Some commands, by the nature of the actions they perform, imply that there are no other data to be entered, others merely state that your input has ended.

Some common Terminators are: RUn, GO, ENd, AGain and REverse.

END

Section 2-10

Format: **ENd (,x)**
x - Target (SHoot) Frame

When the system reads ENd or ENd with a target frame, all input from the Terminal is terminated. If an Indexer is called, the system first performs the index, then prints "READY!" and waits. **Note** that it is ‘looking’ at all the manual switches until you press "Continue". When "Continue" is pushed, the system runs until finished, types "DATA?" and the, once again, waits for you to push "Continue". When "Continue" is pressed, the system returns to read the Terminal.

RUN

Section 2-11

Format: **RUn (,x)**
x = Target Frame

When RUn is read by the system, and if one of the indexers was called, the system indexes and then immediately executes (RUNS). When finished, it types **"RUN COMPLETE Current Frame xxx, DATA?"** and returns to the Terminal for you to enter more data. RUn is just like ENd except that it does not wait, before and after shooting, for you to press "Continue", but it types the system CUrrent frame at the completion of the RUn.

GO

Section 2-12

Format: **GO (,x)**
x = Target Frame

Go is exactly like RUn except that the current frame is not typed at the end of the run.

EXAMPLES:

1) *Shoot from 100 to 120 then to 150 to 250 then 300*

Reposition artwork at frame 250 by hand

ZERO	*(Not Necessary)*
CU,100	*Tell the System the CUrrent Frame*
GO,120	*Tell the System to shoot to 120 without waiting*

(The System Runs - Nothing is typed by the system)

SHOOT,150	*(Set a target frame)*
RUN	*(Could have entered RUN,150 without having to enter SHoot 150)*

(System Runs)

RUN COMPLETE Current Frame 150 DATA? *(System Response)*

END,250	*(Shoot to 250 and wait before and after shooting)*
READY!	*(System Response)*

- At this point, the System is waiting for you to push "Continue", and it is also looking at the hand control switches so that you can reposition manually.

- Operator pushes "CONTINUE" -
System Runs

DATA? - *(System Response)*

- Operator pushes "CONTINUE"

GO,350

System Shoots to 350 and returns to Terminal

2) *Shoot from 1 to 100 - wait before & after Shooting*

CU,1

SH,100

END *(Could have typed END,100 without typing Shoot,100)*

3) *Shoot from 100 to 200 without waiting or printing frame count at the end of RUn - The system already "KNOWS" the current frame is 100*

```
GO,200      or      SH,200      or      CU,100      or      CU,100
                    GO                  SH,200              SH,200
                                        GO                  GO,200
```

All of the 3 commands above will do the same thing

Implied Data Terminators

Section 2-12b

Several commands imply that you wish to END DATA input. Please refer to the individual commands for a description of each. (See System Data Terminators)

All of the following terminate in the same state as if you had typed "END":

REverse
ROtate
AGain
HOld
LEader
ERewind

All of the following commands terminate like a "GO" command:

WInd
GRewind
WWait

OTHER TERMINATING COMMANDS

WIND

Section 2-13

Format:	**WInd,x** x = The frame number relative to the CUrrent frame to which you wish to wind.

When a WInd occurs, the System will CAP the shutter and set the camera motor to advance. It will then look at the system CUrrent frame and the frame number you entered. From this, it decides whether or not to reverse the camera motor. (**PLEASE NOTE THAT THE SYSTEM WILL ALWAYS ASSUME THE CAMERA MOTOR WAS IN A FORWARD DIRECTION WHEN THE WIND COMMAND IS EXECUTED**.) The system then runs to the indicated frame number, sets that number as the CUrrent frame, UNcaps the shutter, and if it reversed the camera motor, it will reset it back to the original direction and then return to the Terminal for further instructions.

EXAMPLE: (**The System Current Frame = 100**)

WI,200	*(Caps, Runs forward to frame 200 and UNcaps)* *(Current Frame now 200*
WI,150	*(Caps, Reverses, Runs to 150, Reverses (Back Forward) and Uncaps Current Frame now 150)*

WIND THEN WAIT

Section 2-14

Format: **WWait,x**
x = The frame to which to Wind.

Wind and Wait is exactly like Wind except that when the frame number entered is reached, the System WAits and looks at all manual switches until "Continue" is pressed, and then returns to the Terminal.

MANUAL SWITCHES

Section 2-15

General Information

The System monitors a group of manual switches at various times.

Appendix "H" contains a list of switches and the time when they are monitored.

The System monitors these switches to direct the flow of execution while the System is running, or to allow you to manually position motors or enter data to the System if in a "waiting" state.

Manual Hand Controller

The hand controller is used to position motors manually. A series of buttons (labeled according to the motor and direction controlled) are used to "tell" the System, the motor and direction you wish to run. Another button located on the side of the box housing the axis buttons, causes the motors to run at a very slow speed when held. This "inching speed" button allows precise manual positioning which could be difficult at normal speed.

When one of the axis buttons is pressed, the system begins to run the indicated motor in the selected direction. The motor is accelerated from a standstill to a "dwell speed". The motor will run at this "dwell speed" for approximately 1½ revolutions. If you keep pressing the button, it will then accelerate to full speed and continue to run until you release it. When the button is released the motor will decelerate to a stop. Once a motor is selected and starts to run, it will continue to run as long as **any** button on the hand controller is pressed before the previous button is released. There is a toggle switch marked "CONT". This switch, if turned on by itself, will run to Zoom in an up direction. To use the "CONT" switch with any other axis and direction, first press the desired pushbutton, then while holding the pushbutton, switch on the "CONT" toggle, then release the pushbutton. The selected motor will continue to run until the "CONT" switch is turned off.

If more than one axis button is pressed at a time, the System will run the axis indicated by the button it sees first. If more than one button is seen by the System as being on at the same instant, then the computer will ignore all the buttons until they are released. From this, you can see that **you can only run one motor at a time from the hand controller**.

When the hand controller is being run, the system keeps track of the positions as they change so that it "knows" where everything is at all times.

The manual hand controller is active only when the System is in a "Waiting" state (See WAIT), and at times when the System displays the READY!, or DATA? message.

MANUAL INDEXERS

(See Indexers)

Section 2-16

There are a series of switches marked for each of the 5 "LOCK" and "INDEX" positions and another marked "RECORD". When the System is in a waiting state (see WAit), these switches are active, and when one is pressed, the indicated action takes place. A position is "LOCKED" into the indicated indexer, or the System is directed to run to the indicated indexer position, or a position is "RECORDED" into the next SAfe location. In any event, the dial counter readings and the name of the INDEXER, LOCK OR SAfe, is displayed on the Terminal by the System to serve as a record of what was done.

MANUAL FADE DISSOLVE UNIT
(See Fade/Dissolves)

Section 2-17

The manual fade dissolve unit provides a convenient method of entering a fade or dissolve to the System, without having to type the data in.

The manual fade/dissolve unit is active anytime the System is in a Waiting state.

When a system terminator command is sent, the System "looks" at these switches just prior to running. This is done even though the terminator itself does not cause the System to look at all the manual switches.

If the switch data on the unit is valid, then before running, the System will type: F/D and the length of fade or dissolve on the Terminal, and then enter the data to the System as though it was typed in.

It is **critical to understand** that the System will enter the Fade or Dissolve starting at the System CUrrent frame at the time the switches are read. Therefore, the System CUrrent frame should not be 0 unless a stop frame (see STOP) has been entered.

A "Step Program" switch is provided and will illuminate when valid data are read from the switches. If the System is in a "Waiting" state and Step Program is pressed, then the System will automatically step through the program data in memory from the CUrrent frame to the frame number equal to the CUrrent frame plus the length of the Fade or Dissolve entered. **Please note** that if any movements have been programed over these frames, they will also occur. Step Program is intended primarily for use when it is desired to use the fade/dissolve unit in a totally manual mode. However, no harm is done in other modes.

PLEASE NOTE:

1) If "step program" is pressed when no valid fade/dissolve data is present, nothing happens.

2) If "step program" is pressed and a manual dissolve OUT was entered, then the System will automatically rewind to the head of the dissolve out when finished. **BUT THE CURRENT FRAME WILL NOT BE RESET**. That is, the CUrrent frame will be that as represented by the end of the dissolve.

Naturally, the System inspects the current position of the shutter to see whether the data on the switches is valid before entering it. As a result, the System will not enter a manual Fade Out if the shutter is already closed, etc.

If a fade or dissolve is entered to the System, and then, before "Continue" or "Step Program" is pressed, you wish to remove it, simply set the IN-OUT switch to the straight up or neutral position. When this is done, the "Step Program" light will go out and the Fade or Dissolve selection is erased. To change the length, Fade or Dissolve selection, or In or Out selection, simply reset the switches as desired. The System will report changes to the Terminal.

If the System is in a waiting state, and neither the in or out position of the select switch seems to work, push the Open-Black switch to be sure that the shutter is not partially open or closed. A manual fade or dissolve will not be entered unless the shutter is in either a full open or full closed position, (unless
OVer is used, please refer to the OVer section of this manual for details).

Any Fade or Dissolve compensation or overlap entered will be applied to the Fade or Dissolve. (See FComp, DComp).

SHUTTER OPEN-BLACK

(See OPen, BLack)

Section 2-18

This switch controls the Dissolving shutter, and is active when the system is in a waiting state.

Pushing the switch toward "Open" causes the shutter to open and the message "Opened" to be printed on the Terminal.

Pushing the switch towards "Black" causes the shutter to close and prints the message: "Closed" on the Terminal.

ACCURIZE

(See ACcurize command)

Section 2-19

The Accurize button is active when the System is in a waiting state. When the button is pressed, **if** the **last motor run by the hand controller** pushbuttons was last run in a counter-clockwise shaft direction, then that motor is run 100 steps further in that direction, then run back 100 steps to eliminate mechanical slack that might exist in the machine being controlled. If the condition mentioned does not exist, the "ACCURIZE" button has no effect.

CONTINUE

Section 2-20

The "Continue" switch is active under several different conditions. In all circumstances, it does just what it says. If the "Continue" switch is pressed, the System is told to continue or "Go Ahead" to **its** next logical task.

For example, if a mechanical limit is sensed while the System is running, a message is typed and the System then "looks" at the "Continue" switch. If pressed, the System will go back to what it was doing when the limit occurred.

In another instance, when the terminator ENd is sent by the operator, the System will wait, looking at the switches. If "Continue" is pressed, it will stop looking at switches and continue on to execute the job entered.

REVISE

Section 2-21

The Revise switch is inspected when the System is in a wait state, when running from a transfer file, and just before exposing each frame of film.

When the system detects the "REVISE" switch, the message: PROGRAM SUSPEND FOR REVISION will appear on the Terminal, and the System will read the Terminal for data to be input. This allows the operator to revise his instructions even after the System has started running.

NOTE: It is possible for the Revise signal to be seen by the System after a frame has been counted but before it is actually exposed. Therefore, it is advised that you pay attention to the CUrrent frame count displayed by REvise. If the count displayed does not agree with your mechanical counter, then you should **usually shoot a frame manually** to "resync" the two.

Resetting the Current count will not suffice if movements are occurring. This is because the frame count advanced by the System, is in the position where that that frame needs to be exposed. If you merely reset the frame count (CUrrent), the movement for that frame will be repeated.

Naturally, if you are going to start over or ZEro the CUrrent frame, none of this applies.

INDEXERS

Section 2-22

General Information

The Cinetron System is "AWARE" of the position of each axis motor relative to the System Center position. This position (Center) is considered to be Zoom 1000, and all other axes at 00. (See Locate Center).

Anytime you move an axis, whether with an indexer, the hand controller, a programmed move, or other means, the System keeps track of that movement.

When a position is "Locked", the System copies the current position **relative to center to memory**.

If you call an indexer to run, it returns the motors to the position it was in at the time it was "Locked". **It is important to note that Indexer positions are copies of the current center position, and if the center position is Reset, then <u>all</u> indexer positions will be affected.**

EXAMPLE

Say the physical Zoom counter was at 1000 when the center was located and you move to 2000 telling the System to lock that position in an indexer.

The System will "Remember" the center position + 1000 is that position.

If you LC (Reset or Locate Center) without moving from the 2000 Zoom counter position, the system internally sets this position telling the indexer to run. It will then run to 3000 which is the center position + 1000.
This characteristic relationship between the absolute center position and all the Indexers can be used to your advantage in some cases.

EXAMPLE:

If a job is setup and shot, and then it is decided that everything is ok except that all action should take place 1 field north of the original, the center can be reset 1 field north of the true center and then the same commands can be used as in the original. Thus, all action will be duplicated 1 field north of the origin.

NOTE: The counter readings printed by the System will be off on the N/S axis

by an amount that equals 1 field.

LOCATE ABSOLUTE CENTER

(See Indexers)

Section 2-23

Format: **LCenter**

The System keeps track of the positions of all axes based on an absolute Center position which you can set. The System considers that the absolute center is a position where all axis counters on your machine are at 0000 except for the Zoom Counter which is considered to be at 1000.

The procedure for resetting the Absolute Center is as follows:

1) Move the Zoom AXIS until its counter reads 1000

2) Move all other axes until their counters all read 000.

3) **Type LC** (Locate Center)

The system will request that System Security Code to prevent accidental relocation of the Center position.

Enter the System Security Code (The Default is OKGO).

4) If the security code is valid, the System will respond:

RESET ABSOLUTE CENTER!

If the security code is not valid the message will not be printed, and the center position will not be reset.

It should be noted that at certain times the System will display what it considers the various counter numbers to be.

These counter readings are dependent on two things:

1) The Scale of the axis (See Scaling, DScale, VEeder)

2) The actual point at which the Center was located

If the Zoom Counter actually read 2000 when LC was entered, then all printouts of the Zoom position would be 1000 numbers too low. This is because the System "thought" that the Zoom was at 1000 when the Center was reset.

It should also be noted that the Indexers are all set relative to the absolute center position, and **when the absolute center is reset then all the indexers will be affected**. (See Indexers)

LOCK

Section 2-24

Format: **LOck**

When an indexer is "Locked", it is told to remember the current position of all the motors in the System. The CUrrent frame count is **not** a part of an indexer, **nor is the position of the Fade/Dissolve shutter**.

In Appendix "E" there is a list of command identifiers for each of the indexer's "LOCK" commands. Each "LOCK" command refers to a specific "indexer" command. That is, "1L" (or one lock) points to 1I (or one index), and so forth.

The System may be told to "lock" an indexer either by typing the applicable command identifier via the Terminal, or by pressing the proper lock button when in the Hand Controller mode.

INDEX

Section 2-25

Format: **INdex**

Appendix E contains a list of command identifiers which are used to "call" and indexer when in the Terminal mode. If an indexer is called via Terminal Commands, no immediate action takes place. Rather, the indexer is made active when a Terminator command (See Terminator Commands) is typed. The indexer will then run to the position the machine was in when its lock was typed.

Indexers may also be commanded to run by pressing the appropriate button when in the Hand Control Mode.

EXAMPLE: *The axes are run to ZOOM 1000, NS 100 and 1 Lock is typed or the 1Lock manual indexer button is pressed. At this instant, the System memorizes all axis positions in the 1 Lock Memory.*

The motors are run to ZOOM 2000, NS 800, EW 200, then the following is typed:

ZEro

CU,1

SH,10

1Index - *Command to call an indexer may be written any time **before** the terminator.*

North,100,1,10,1,BO -*Axis movement call*

END -*terminator*

- *System Response: Indexer will run to the 1 Index Position **AS SOON AS A TERMINATOR IS TYPED**. In this case "END" and system will type "**READY**" and wait for 'continue' to be pressed.*

READ THIS!

When Continue is pressed the N/S pan will occur and film will be exposed.

DATA? *System requests data at end of job*

6 INDEX

Section 2-25b

Format: **6Index**

The 6 Index is a reserved "Housekeeping" indexer.

Anytime the System receives a command which will cause it to run, (a terminator) it does an automatic 6 Lock. (See LOck)

The 6 Index command is made available to cover those occasions where the operator needs the head of a move "locked" in an indexer but forgot to tell the System. If this happens, the System will have remembered, in the 6 Lock, where everything was when the **last terminator** was given. A command to 6Index will return the System to that position.

The 6 Lock occurs when the terminator is typed and before any indexers are run.

EXAMPLE: *The axes are at Zoom 2000, NS 100*

ZEro

Up,1000,1,10,1,BO

CU,1

GO,10

At this point Zoom 2000, N/S 100 are placed in the 6 Indexer (6 locked)

The System runs up 1000 units & shoots 9 frames

Now the position is Zoom 3000, N/S 100

At this point, the operator discovers that he wants to repeat the last move, but he forgot to lock the position of the beginning.

6I -	*Command to Return to point where last terminator was typed (GO,10)*
AGain -	*Terminate and Repeat*

- Now the System returns to 2000, N/S 100

READY!	*- System Responds*

Shoots & Repeats the move when operator presses "Continue"

DATA?	*System requests more data*

IMPORTANT NOTE!

The 6Index is used by the System while executing the WRite, SMulti, DRaw, CYcle, SCan, MUltie Direct Move, FIgure, and MFigure routines, so its housekeeping feature will not work following execution of one of the listed routines...that is to say typing 6Index immediately following one of these special routines will not return you to the position you were in when you typed the terminator to start one of the special routines.

IOFF

Section 2-26

Format: **IOff**

The IOFF command removes all index commands from the active list. That is, if you "called an indexer" then IOFF removes the call. Typing IOFF will cause the System to cancel all commands to run **any and all indexers** –

The System will respond: "**ALL INDEXERS OFF**" when IOFF is received. IOFF does not affect the positions memorized, just the commands to run to these positions (Index, 1Index, etc.)

EXAMPLE:

An instruction to 4Index was written. Then it was discovered that 3Index, not 4Index, was the actual position desired.

-

- *(various commands)*

-

4Index

-

- *(more commands)*

-

IOFF	*-Command to turn off all indexers*
All Indexers OFF!	*System response*
3Index	*Turn on 3Index*
GO	*-Terminator*

System will run to 3Lock position before shooting

AUTO

(See AUdition, Indexers, Expose, AFwd, ARev, Automatic Moves)

Section 2-27

Format:

AUTO (**4 letters are required**)

The AUTO command locks the current position to the main Index position and places the System in a WAit state where the hand control is active. An Index command will return the axis to this position.

EXAMPLE:

```
LOCK
        = AUTO
WAIT
```

-----and-----

```
        LOCK
AUTO =
        WAIT
```

AUTO cannot be shortened to 2 letters since AU is the command identifier for AUdition. If AUTO is misspelled, the System will go into AUdition mode, and the hand controller will not be active. IF this happens type EXpose to remove AUdition, the type AUTO properly.

SECTION 3

SAFE, TRACKING AND CENTER COMMANDS

Section 3-1

General Information

Cinetron Systems report axis positions to the operator at various times. It reports the positions in numbers which represent the readouts of non-resettable, one-direction, mechanical counters. The System assumes that all these counters were reading 0 (except for the Zoom which is assumed to be reading 1000) when the Locate Center command was entered (See LCENter). A one-directional mechanical counter will count from 0 to 1 to 2 etc. when rotated in one direction and from 0 to 9999 to 9998 etc. when rotated in the opposite direction. Most animation stands and modern optical equipment is equipped with this type of counter. A potential ambiguous situation exists regarding these counters for numbers greater than 5000. This is owing to the fact that 6000 may either be 6000 counts in one direction or 4000 in the opposite direction. Your System resolves this by **NOT ALLOWING A NUMBER LESS THAN ZERO ON THE ZOOM AXIS**, and by allowing all axes to be scaled (See DScale and VEeder) to avoid this ambiguity. Therefore, a number smaller than 5000 will imply one direction and a number greater than 5000 will imply the opposite direction except on the Zoom axis where all numbers are assumed to be above 0.

When the System reports a position, it always uses the same format (display order) in its printout. The order used by the System on printout is the same as is used to enter a position for the TRack,CEnter, ULock, PStep, RPos, NStep, RNeg commands (Command IDENTIFIERS). The Format used is as follows:

Command identifier , Zoom Axis, N/S, E/W, Rotation, P1, P2, P3

If the auxiliary axes (greater than 8) are included in the System then the next line displayed or printed will be: **CONT*, Peg 4, AUX 1, AUX 2, AUX 3, AUX 4, AUX 5, AUX 6**

NOTE: The CONT* indicates the printout is a continuation of the first line and is <u>not</u> entered by the operator.

The System will always print at least the Zoom axis, if all positions to the right of the last Non - 0 position are 0, they will not be printed, if not, 0 will be inserted as a placeholder for each axis until the rightmost non-zero entry is printed.

EXAMPLE:

Zoom 1000 NS 0 EW 0 Rotation 100 and all other position = 0 would be:

(Identifier) 1000, 0, 0, 100

The first number being ZOOM axis, the next North-South, the next East-West, the next positions Rotation - Since the Rotation (100) was the last printed all others are at 0, and are not printed.

SAFE

(See RECOrd, Indexers)

Section 3-2

Format: **SAfe,The number of the desired safe location**

(Safe locations are numbered from 1 to 99.)

General Information

The safes are not unlike the indexers, in that they save the locations of all the axes where either RECOrd was typed or when the manual Record button was pressed. **They differ from the indexers** in that they save the locations **TO THE NEAREST WHOLE COUNTER NUMBER**, where the indexers save the positions to the motor step which permits a fractional position.

For example if, when the Record button is pushed, the zoom is at 1000 + ½ then 1001 will be saved. If the zoom is above 999 ½ and below 1000 ½ then 1000 will be saved.

It is **important to remember** this when using the SAfes to return to an exact position as there is a potential ½ number difference between the Recorded **Safe** position and the true position.

Depending upon the Scale (See DScale, VEeder) of the axis this difference can usually be ignored unless many matched passes are to be made at a relatively tight field since one half a number is usually only five thousandths of an inch. When it is desired to make multiple runs from a recorded safe position it is a good practice to tell the system to run to that safe position before making the initial exposure. This will insure that if there was a difference between the actual and recorded positions, then this difference would be cancelled since the first and subsequent runs all start from a run to SAfe position.

Using a SAfe Position

When the SAfe command and a SAfe number are read by the System via the terminal the System **immediately runs to that position** and returns to the terminal for more input from the operator. SAfe positions are not affected by zero.

EXAMPLE: We wish to run to the position recorded in SAfe 10.

SA,10 *Command*

The system runs the motors to the position where they were when this position was RECOrded.

WAIT AT SAFE

(See SAfe)

Section 3-3

Format: **WSafe**

The WAit and SAfe is exactly like the SAfe command except that after running to the indicated position the System looks at the hand controller switches and waits for the operator to push the "Continue" button before returning to the Terminal for more commands.

RECORD

(See SAfe, REverse, REstore, COunter FOllow, LSafe, ?Protected)

Section 3-4

Format: **RECOrd**

Notice that the four letter sequence RECO is required. When RECOrd is typed, the System responds with the newly recorded SAfe number, followed by the counter readings for each axis in the standard format (See SAFE TRACKING-CENTER COMMANDS General Information).

RECOrd may also be accomplished while in the hand controller active modes by pressing the button marked RECORD.

RECOrd records the current position of all axes in the next highest SAFE number unless 99 was the last safe recorded in which case it will restart at Safe location 1 unless it lies within a protected range (See LSafe). When the RECOrd command "wraps around" past SAfe 99 it will be replacing the information in each SAfe as it is RECOrded. The original SAfe positions are lost.

LOCK SAFE AREA

(See RECOrd, ?Protected, SAfe)

Section 3-5

Format: **LSafe, x, y**
x = A safe number from 1 to 99 (or 0 to clear)
y = A safe number higher than x (or 0 to clear)

LSafe is used to protect a range of SAfe locations to prevent the accidentally recording over them. When the LS, x, y is read by the System it will respond:

ENTER YOUR NAME PLEASE

At this point you may enter **any 7 characters or numbers** you wish. Preferably something to help you identify why the area was protected should be used.

The system will then protect or "lock away" these locations and will not record over them unless a subsequent LSafe command is entered.

The LSafe area is not affected by ZEro.

ONLY ONE AREA MAY BE PROTECTED AT ANY GIVEN TIME.

EXAMPLE:

We have recorded 30 positions in Safes 10 through 40 which we wish to protect from being overwritten.

LS,10,40	*- Lock Safe Request*
ENTER YOUR NAME PLEASE	*System Responds*
EXAMPLE	*Your Reply*

At this point the System will not record new information in SAfes 10 through 40 and will identify this area by the name "EXAMPLE".

If an attempt is made to RECOrd over a protected (locked) safe area the system will respond: (using the first example)

SAFE AREA 10 to 40 PROTECTED BY EXAMPLE NEXT SAFE OPEN 41

and in this case RECOrd next in SAfe 41 the system also types:

SAfe 41 Zoom# NS# EW# etc.

To clear the locked safes enter:

LS,0,0

The System will then unprotect any existing protected range of safes.

DETERMINING PROTECTED AREAS

(See LSafe, SAfe, RECOrd)

Section 3-6

Format: **?Protected**

?Protected is typed to the System so that the operator may determine what range of safe locations are protected, and by whom.

If none are protected then the System response to ?Protected will be:

NONE PROTECTED

If an area is protected then the System will respond:

x to y Protected by zzzzzzz

Where x = lower safe #

y = upper safe #

zzzzzzz = up to 7 letters or digits used as identification

EXAMPLE: *We have protected or "Locked" safes 10 through 25 and identified them by the name TEST 10.*

If we type?

?P

The System will respond:

10 to 25 protected by TEST 10

COUNTER

(See RECOrd, SAfe, LSafe)

Section 3-7

Format: **COunter, safe number from 0 to 99**

COunter is used to reset the RECOrd SAfe counter to force the System to RECOrd **next** at a pre-determined SAfe. When the COunter command is received the System will respond with the next available safe number. If 0 is entered as the COunter number, the System will interpret it as 1 since 0 is not a SAfe number.

EXAMPLE: *We wish to start RECOrding at SAfe 20*

CO,20	*Request to start at 20*
NEXT SAFE OPEN 20	*System responds*

At this point, the System is set to RECOrd into SAfe 20 **the next time** that RECOrd is typed or the RECORD button is pressed if in the hand controller mode.

NOTE: If the SAfe number lies within a Protected Range (See LSafe) then, when a RECOrd command is received, the System will record at the first SAfe location beyond (higher numbered) the Protected area.

FOLLOW

(See SAfe, RECOrd)

Section 3-8

Format: **FOllow, x,y**
x = SAfe number to Start following positions
y = SAfe number to Stop following positions

FOllow is used to shoot in each SAfe position from the starting SAfe number position to the Stop SAfe number position entered. Primarily FOllow is useful in photographing rotoscoped scenes or to follow free hand drawn curves, etc. Follow will shoot 1 frame in each SAfe position unless the HIt command is used to shoot more than 1 frame.

FOllow requires that a current frame be entered.

FOllow will shoot until at least 1 frame has been shot in each position from the SAfe start to the SAfe stop positions. **The target frame (SHoot) is meaningless**.

EXAMPLES:

#1*) A series of positions have been rotoscoped and RECOrded in SAfes 21 through 81. We wish to shoot 1 frame in each position.*

CU,1 *Some current frame must be entered*
FO,21,81
GO

At this point the system will shoot 60 frames; 1 in each position from SAfe 21 to SAfe 8

#2*) The same as Example #1 except that we wish to shoot from Position 81 back to 21*
CU,1
FO,81,21
GO

The first exposure will be at SAfe 81 - the next at SAfe 80 then 79, etc., until the last is shot at 21.

#3)*The same as Example #1 except that we want 2 frames in each position.*
CU,1
HIt,2
FO,21,81

GO

NOTE: FOllow may be used to FIgure or MFigure for Series IV Scan. MDirect (Move Direct) has no meaning with FOllow since you could use the SAfe command to run to a particular point within a FOllow.

TRACK AND CENTER

(See CEnter, RECOrd, Indexers)

Section 3-9

General Information

The TRack and CEnter routines accept data in the same format as the RECOrd command prints it. That is, Zoom counter number, N/S counter number, E/N counter number, etc. ---. Please refer to the RECOrd and SAfe sections general information paragraphs

TRACK

Section 3-10

Format: **TRack{,Zoom Destination, N/S,EW,Rotation,Peg1,Peg2,Peg3}**

TRack will cause the System to run to the entered counter reading on each axis as soon as it is typed, and will return to the terminal after running.

A number representing the counter reading must be entered for each axis until all axes to the right of the last entry are to run **to** 0. **If TRack is entered with no numbers the System will run to the System Absolute Center (See Locate Absolute Center)**

EXAMPLE: *We wish to run **to** Zoom 2000 NS 0, EW 100, all other axes 0*

Enter:

TR,2000,0,100

The System will run **to** the indicated **positions** which are:

```
Zoom   2000
NS     0
EW     100
Rotation 0
Peg 1  0
Peg 2  0
Peg 3  0
```

EXAMPLE: *We wish to run **to** Zoom 1000, all other axes 0*
(this is the absolute Center, see LC).

TRac

EXAMPLE: *We wish to run **to***
Zoom 100, N/S 0, E/W 0, Rotation 0, Peg 1 0, Peg 2 0, Peg 3 4920
Enter:

TR,100,0,0,0,0,0,4920

Notice: that it is necessary to enter a number indicating the desired counter position for each "Motor Place" in the format so that the System will realize that 4920 refers to Peg 3

IMPORTANT: Note that the 0 entries mean run until the counter reads 0, they are NOT placeholders in the command.

CENTER

(See TRack)

Section 3-11

Format: **CEnter{,Zoom Destination, N/S,EW,Rotation,Peg1,Peg2,Peg3}**

(This is the same entry format as the TRack command except the identifier is CEnter)

CEnter is exactly like TRack except that when the motors reach position the System “looks at” the manual hand control switches and waits until the operator presses "Continue" before returning to the terminal for more data.

STEP ORIENTED REPOSITIONING ROUTINES

Section 3-12

(See DScale, VEeder)

General Information

The step oriented repositioning routines are included in the System for those users who are **PROFICIENT ADVANCED COMPUTER PROGRAMMERS** and wish to write their own programs on external computers and use the Cinetron System as an intelligent controller.

These routines have very little (if any) application in the normal day to day operation of the System, but were included merely for the purpose stated above.

The step oriented routines (PStep, NStep, RPos, RNeg) are commands to run distances given as **motor steps**.

RESTRICTION - Remember that the largest number recognized by the System is 32767, therefore if more steps are to be run in a single command, you must devise a method so that no number entered will exceed 32767. This value represents about 82 revolutions of a standard Cinetron 400 step motor or about 8 inches on a typical animation table. Rescaling the axis is one approach.

The order of the entries and sense for these routines is the same as TRack or CEnter except that motor steps are used to indicate distance not counter numbers.

POSITIVE STEP RELATIVE

Section 3-13

Format: **PStep {,Zoom Steps, N/S Steps, E/W Steps, Rotation Steps, Peg 1 Steps, Peg 2 Steps, Peg 3 Steps}**

PStep causes the motors to step the number of steps indicated from the current position in a direction which the system considers positive. The same format for entry applies as required for the TRack or CEnter routines **EXCEPT THAT THE DISTANCE MOVED IS RELATIVE TO THE CURRENT POSITION OF EACH AXIS**.

EXAMPLE:

We wish to move the Zoom 100 Steps positive from its current position and the E/W 1000 steps positive and leave other motors where they are now.

Enter

PS,100,0,1000

The "0" was entered to indicate that the N/S axis was not moving **THIS IS NOT THE SAME MEANING AS A 0 ENTRY IN THE TRack and CEnter ROUTINES**. 1000 is the E/W Entry - Since nothing else was entered, the system considered all entries to the right of 1000 to be 0 and thus did not move them from their current positions.

NEGATIVE REPOSITION RELATIVE

Section 3-14

Format: **NStep {,Zoom Steps, N/S Steps, E/W Steps, Rotation Steps, Peg 1 Steps, Peg 2 Steps, Peg 3 Steps}**

NStep is exactly like the Pstep except that the System moves motors in the opposite direction (negative direction).

REPOSITION POSITIVE STEPS ABSOLUTE

Section 3-15

Format: **RPos {,Zoom Steps, N/S Steps, E/W Steps, Rotation Steps, Peg 1 Steps, Peg 2 Steps, Peg 3 Steps}**

Reposition Positive Steps Absolute will move the axes **to the indicated positive position, in motor steps, relative to the ABSOLUTE CENTER POSITION, not the current position. HOWEVER IF AN ENTRY IS 0 THAT AXIS WILL NOT MOVE.**

EXAMPLE: *We wish to reposition the Zoom 120 steps Positive (Up) from the Absolute Center and the N/S axis 100 steps positive (North from the absolute center, and leave all others where they currently are.*

RP,120,100

The System will move to center +120 ***steps*** *on the Zoom and +100* ***steps*** *on the N/S - all other axes are left alone.*

EXAMPLE: *We wish to positions to +120 Zoom +200 N/S with all other axes at 0 and we don't know where we are now.*

Enter: **TRack** *to set all 0 and Zoom 1000*

Now we know where we are (Absolute Center, all axes at 0 except zoom at 1000)

RP,120,200 *Now reposition N/S and Zoom*

REPOSITION NEGATIVE STEPS ABSOLUTE

Section 3-16

Format: **RNeg {,Zoom Steps, N/S Steps, E/W Steps, Rotation Steps, Peg 1 Steps, Peg 2 Steps, Peg 3 Steps}**

RNeg is exactly like RPos except that the data entered are considered to be **negative motor steps** from center.

EXAMPLE: *We do not know where we are but we wish to position to Zoom - 1000 N/S +100, Rotation 100*

TRack	*to run to Absolute Center*
RN,100,0,0,100	*Move zoom and rotation to 1000 and 100*
RP,0,100	*Now move the N/S to +100*

USER INDEX/LOCK

See (AMove, TRack, CEnter)

User LOCK

Section 3-17

Format: **ULock,Zoom Destination{, N/S,EW,Rotation,Peg1,Peg2,Peg3}**

The positions are entered as **counter readings** for each axis. At least one position must be entered, otherwise the command is meaningless. The entries are in the same format as the TRack and CEnter commands and you should refer to these sections of the manual for further details.

The USer Lock is used primarily as a method of "Locking" a positions without having to run the position. User Lock is particularly useful as an Origin or End Index for the AMove routine.

EXAMPLE: *You wish to indicate Zoom,2000, N/S 100, E/W 100 as a user lock position*.

ULock, 2000, 100, 100

Note that all positions to the right of the last entry (E/W in the above example) are set to 0 by default. **Note that at least 1 position (the zoom) must be entered**.

USER INDEX

(See ULock, CEnter, TRack, AMove)

Section 3-18

Format: **UIndex**

When UIndex is typed, the System **immediately runs** to the position indicated by ULock. Notice that the user index runs at the time the command UIndex is typed. **It does not wait for a Terminator** as do the normal indexers. After running to position the System returns to the terminal for further commands.

ZERO REMAINDERS

Section 3-19

Format: **0Remainders**

0Remainders is a command normally associated with the operation of the System from an external computer.

When the system calculates a position it calculates in terms of the number of motor steps to run. Not all calculations will result in an exact number of motor steps. The System resolves this by saving any fraction of a step, for each move command, in a remainder. When a motion command is first entered its remainders are set to 0. As the System runs, this remainder for each axis changes depending upon whether or not the calculations will land exactly on a motor step. The Remainder method insures that the cumulative position of the motors is never more than ½ a **motor step** away from the exact mathematical position calculated. This represents 0.000125 inches or 125 millionths of an inch maximum positioning error when using a 400 step motor or servo.

If it is desired to repeat a movement and AGain, REverse, or ROtate is used, the System will **adjust** (usually reset to 0) the remainders as required to maintain this accuracy and ensure that repeated motions match exactly.

If, however, you wish to repeat a movement by merely returning to the start position and resetting the CUrrent frame count, no adjustment to the remainder is done, and there is a potential error of the amount that the remainder represents. Therefore, if you use 0Remainder, the remainders are all set to 0 again and you start over "fresh".

All of this may be a bit confusing. If it is, just ignore 0Remainders and read on to the next section. Ignore it because in most cases the remainders are so small and represent such a small difference in position that your lenses and film can't resolve the difference unless many, many passes are made over the same section at a relatively tight field.

EXAMPLE: *We shot a scene from 1 to 100 and wish to repeat it.*

CU,1	*Reset frame count*
0R	*Zero Remainders (if you're picky)*
GO,100	*Will reshoot from 1 to 100*

--*OR JUST TYPE*--

AGain	*Do last operation again*

--BECAUSE--

Resetting the CUrrent and SHoot frame and using 0Remainders is exactly what AGain does except that AGain is a terminator and will end data entry and wait until the 'Continue" button is pressed whereas the first entry method will not require operator intervention before running. The system realizes that when you type AGain, the remainders need to be reset. If you typed ROtate, as a command to "back up" over the previous move, the system would reverse the sign of the remainders so that the move would exactly track over the first.

DIRECT SCALING DScale

Section 3-20

Format: **DScale,# steps,motor#**

Direct Scale (DScale) is used to indicate the number of **motor steps in 10 counter indicator counts.** This is to insure that instructions requiring distances to be entered match the distances shown on the machine's indicators.

As mentioned before, the number of steps is the number of steps the motor must take to change the indicator or counter 10 counts. The stepping motors used in Cinetron Systems *normally* are set to require that 200 or 400 steps will rotate the motor shaft 1 time. Therefore, if you rotate a shaft once and the indicator counter attached to the shaft changes 10 counts then the direct scale of the motor is probably either 200 or 400 since most commonly used counters on animation/optical equipment register 10 per revolution of the connected shaft. Most Cinetron systems use 400 step motors.

The motor number is a number in the range of 1 to 14 indicating the motor or axis you wish to scale. See Appendix "C" for a listing of motor numbers.

When the System reads the direct scale, it will respond:

Maximum Single Frame Distance ZZZ

ZZZ indicates the maximum distance that motor can be commanded to move on a single frame. Note that this has nothing to do with the operation of the indexers, TRack or CEnter commands insofar as maximum distances are concerned. This maximum pertains primarily to Slitscan operations. (See LAse).

For maximum accuracy in setting the scale for an axis which is driven through a gearbox or a combination of belts, chains or pulleys, the VEeder command is recommended. (See Section 3-21)

VEeder

(See DScale)

Section 3-21

Format: **VEeder**

VEeder is an interactive motor scaling routine which will prompt the operator for data as the scaling routine progresses.

VEeder is designed to be used primarily on an axis which is being driven through a gearbox or a series of pulleys and is therefore difficult to ascertain what direct scale to use. Upon completion VEeder will type the direct scale value which is closest to the scale which was obtained by running the motors.

The sequence of events for VEeder is as follows:

VE *Operator invokes veeder command*

ENTER SECURITY CODE *System ask for security code*

URUN *Operator enters his security code (Security code for your system is in a separate document - ask the owner of the System)*

SCALING IN PROGRESS *System response to legal security code*

(System now runs to center 1000,0,0,0,0.........0)

WHICH MOTOR NUMBER? *System requests motor # to run. See Appendix "C" for motor #'s*

1 *Operator enters the number for the axis he wants to scale (Zoom in this case)*

OK-RUN THE SELECTED MOTOR *System response*

At this point the entire hand controller is active, including the indexers. Any motor or combination of motors may be moved, but **only the selected motor will be scaled**. When the selected motor (zoom or #1 in this case) is in position and its counter reading noted – push the "Continue" button the system responds:

ENTER THE AMOUNT THE COUNTER CHANGED - *The System wants to know how much the counter on the selected motor changed from the time the System said to move it. If the zoom started at 1000 and now reads 3000 then enter:*

2000 *Enter the amount the counter changed*

LARGEST SINGLE FRAME MOVE "xxx" *System response*

CLOSEST DIRECT SCALE "YYY"

"xxx" indicated the greatest distance that the System can move between 2 frames (See DScale)

"yyy" indicated the number that is the closest number for the ***DScale*** *routine.*

When scaling is finished the System will run to the positions the motors were in when VEeder was typed.

AUTOMATIC MOVES

Section 3-22

General Information

Normally commands to move an axis require that each axis to be moved be listed separately along with the direction and distance to run.

The automatic moves allow the System to figure out the direction, and distance based on what you define as the origin and end of the move you wish to make.

Distances for automatic moves are calculated on the basis of whole counter digits. That is, it is not possible to make an automatic move of 100½ numbers any more than you can instruct the System to move East 100½ numbers. It is important to note that this characteristic exists because the Origin or End of an automatic move is defined by an indexer position and or the current position. Both of these have much higher resolution than counter numbers. Therefore, **an automatic move will be calculated between the nearest counter number to the origin and the nearest counter to the end**.

EXAMPLE: If 100½ is the origin position and 200¾ is the end position for a given axis (in counter values), then the difference calculated will be 101 numbers. When the move is made from 100½it will finish at 201½. This degree of accuracy is more than adequate for any but the most critical work. (One digit is typically 1/100 inch)

If it is desired to increase this accuracy, the System should be given a command to TRack to the origin rather than using a call to an indexer to go to the origin.

In the example just given, if the System is told to TRack to 100 (the closest counter # to the origin) the move will begin at 100, exactly then move 101 units and end at 201 which is only ¼ of a number off the exact position.

Accuracy can also be increased by rescaling the machine so that the counter readings are 2, 5 or even 10 times larger, (See DScale, VEeder). If this is done, it should only be done in those rare cases where it is absolutely required, and then the System scale should be reset to its normal value when finished.

Many Optical Printer systems controlled by Cinetron use higher resolution counters.

AUTOMATIC FORWARD

Section 3-23

Format: **AFwd, Start Frame, Stop Frame, EaseIn Frames, EaseOut Frames**

General Information

The Automatic Forward and Automatic Reverse routines both use the main Index and Current position to determine what the direction and distance of the given move are to be. That is, the Current position of all axes and the Index position of all axes will each be one end of the movement calculated.

To set one end of the movement use the AUTO or LOck command. This is the position to which the System would run if INdex were entered.

Any number of automatic moves may be given, but you should know that each axis which has an automatic move uses one of the 8 positions described in the **Axis Movement Conventions section** of this manual.

When AFwd is entered the System will assume that every axis which is not currently on its INdex position is to have a move entered from the INdex position to the current position. If AFwd is typed while all axes are on the index position an error is assumed and the System will **IGNORE** the entry. This is because if the end and origin of the move are both on the same position then no motion can occur.

Note that **every** axis that has been moved even the least amount from its index position will be entered as an automatic move with the start move, stop move, taper in and taper out as given in the AFwd command. The instructions for each axis involved are written automatically to the System just as though you had entered each axis command separately in a conventional manner. This is why the system uses whole numbers as the move distances as discussed in “General Information” section 3-22 on automatic moves.

Please refer to the General Information discussion of the AXIS MOTION SECTION for a discussion of the 8 instruction level buffer used for each axis. If only the zoom axis is moved off its index position, using an AFwd command would be exactly like writing a UP or DOWn command with the same description. If 2 axes are moved one entry will be made for each of the two axes and so on.

The AFwd command will automatically set the INdex to run. If you don't wish the System to run to the Index position, use IOff to turn off all the indexes before typing a terminator. IOff will prevent any indexer that is scheduled to run at the time it is typed. IOff will not prevent any subsequently called indexer from running.

To enter another move as an AFwd (or ARev) simply re-LOck the origin of the next AFwd (or ARev) move desired.

EXAMPLE: *We wish to zoom up from 1 to 100, at frame 50*

We wish to add an East/West movement ending at frame 100

1Lock

LOck

(These 2 commands sets 2 indexer positions at the same point)

Move to the origin of the move

AFwd,1,100,5, Both *(Describe Zoom, it will run from the Index point to the current position)*

LOck (or AUto) *Set the current position as the origin of the East-West axis (you can ignore the other axes)*

Move the East-West axis to its end position

AFwd,50,100,6,5 *(Describe East-West move)*

IOff *(We do not wish to start at the Index position since it now describes the beginning of the East-West but also the end of the ZOOM.)*

1Index *(Instead we want to return to the origin of the whole move which we set as 1Lock when we started. Since 1Index was typed after IOff, the system will run to the 1Index position*)

CU,1

GO,100

The System will run to the 1 Index position and then shoot the move as we wanted

AUTOMATIC REVERSE

(See AFwd)

Section 3-24

Format: **ARev**, **Start Frame, Stop Frame, EaseIn Frames, EaseOut Frames**

ARev is exactly like AFwd except that it considers the LOck position to be the end of the move and the current position to be the beginning. Because of this, **ARev does NOT automatically set the INdex to run.**

VARIABLE AUTOMATIC MOVES

(See Automatic Moves, AFwd, ARev)

Section 3-25

Format: **AMove, Start Frame, Stop Frame, EaseIn Frames, EaseOut Frames**

The Variable Automatic Move is very similar to AFwd and ARev except that it allows you to specify the origin and the end of the moves instead of defaulting to the Current position and INdex position.

The origin index and end index positions are each defined by the operator and may be changed at will.

The **Variable Automatic DOES NOT AUTOMATICALLY RUN TO THE ORIGIN TO THE MOVE**.

ORIGIN INDEX

(See AFwd, ARev, SAfe)

Section 3-26

Format: **OInd**ex**,x**

x = The identification for the INdex or SAfe to be used as the origin of a Variable Automatic Move. (See Appendix "F" for the identifiers required.)

This identifies the **origin of a move** to the System to be described with the AMove command. The identification refers to the positions of each axis stored in the referenced indexer or SAfe **AT THE TIME THIS COMMAND IS TYPED**. **Please note that if the values in the referenced indexer are changed after the OIndex command is issued (re-LOcked), those changes are NOT recognized as far as the AMove routine is concerned**.

END INDEX
(See OIndex)

Section 3-27

Format: **EIndex,x**

x = The identification of the INdex or SAfe to be used as the end of a Variable Automatic Move (AMove).

EIndex is exactly like OIndex except that the **end of the move is defined** not the origin.

AMOVE (VARIABLE AUTOMATIC)

(See OIndex, EIndex, AFwd, ARev)

Section 3-28

Format: **AMove, Start Frame,Stop Frame,Ease In,Ease Out**

The AMove command is exactly like the AFwd or ARev routines except that the movement is calculated from the OIndex position to the EIndex position and **NO AUTOMATIC INDEXING to the origin of the move will occur.**

EXAMPLE: *SAfe 6 has the beginning of a move and INdex has the end position. We want to move from SAfe 6 to INdex, from Frame 234 to 416 with 5 frames tapering in and 10 out.*

OInde,S6	*Set SAfe 6 as origin (unless already done)*
EInde,I	*Set index as end*
. . .	*Other commands as necessary or desired*
CU,200	*Set a Current Frame as desired*
.	
AMove,234,416,5,10	*Set in the move*
SA,6	*Run to SAfe 6, the origin of the move*
GO,416	*Then shoot*

MOVE DIRECT ROUTINE

Section 3-29

Format: **MDirect, Target Frame Number**

The Move Direct Routine is used to reposition all the System motors and the CUrrent frame **count** to the positions indicated by the direct move target frame. **No film is exposed by MDirect routine.**

EXAMPLE: *If a move is entered between frames 1 and 90, and it is desired to run to the position the System would be in at frame 70 of the described move(s).*

CU,1	*Tell the System where to start looking for a move*
MD,70	*Tell the System the target frame of the Move Direct*
GO	*Terminator (MDirect has set the target as 70*
MOVE DIRECT MODE	*System announces mode*
CURRENT FRAME 1	*and current*
TARGET FRAME = 70	*and target frames*

The System will then move to the indicated positions and when done the current frame will be 70.

NOTE: You must be in the correct position for the initial CUrrent frame entered so it may be necessary to use an indexer before entering MDirect.

AND notice that MDirect also finishes with the system Current Frame set the same as the move direct target frame.

STROBE ROUTINES

(See also PAuse, ALl)

Section 3-30

Format: **XI{or XO or XB},Length of fade, {spacing, offest, # of colors to use.}**

The STROBE routines are used to create fading trails on any given move. The trails can be fading in or fading out or both fading in and out.

Operation:

The "STROBE" call is usually entered at the end of a command sequence prior to the entry of a data terminator.

The STROBE identifiers are:

(i) **XI** for fading-in a STROBE

(ii) **XO** for fading-out a STROBE

(iii) **XB** for fading-in and out a STROBE

The use of these strobe routines requires a valid CUrrent frame, target frame (SHoot) and length of effect in frames.

FORMAT for entry:

{XI,XO,XB} Length of fade, (spacing, offest, numbers of colors to use.)

EXAMPLE: Assuming a CUrrent frame of 1, and AFwd, 1, 24, 1, BO being entered in this manner:

XI,12

END,24

The computer then comes up "READY" and shoots the move with 12 additional fading moves, **which give a total of 35 frames**.

The entries for spacing, offset, and colors are all optional, but if any one is used at least 1 must be entered for any skipped entry.

EXAMPLE: An 8 frame strobe out using 15 colors on the wheel would be entered:

XO,8,1,1,15

The 1, (1 for spacing and offset) must be entered (See below)

Spacing Entry - The spacing is the number of frames between exposures - normally either 1 or the same as the offset entry.

Offset Entry - The offset is the number of frames to advance CUrrent count between the beginning of each strobe effect. Normally 1 or the same as spacing.

EXAMPLE: *A front run of a solid object goes through a 100 frame move, it is then desired to make a second strobe pass which will strobe out 8 frames at every 4th position of the original move.*

The entry would be:

OPEN	***INITIALIZE SHUTTER OPEN***
CU,1	*This is*
AFwd,1,101,12,6	*the*
SH,101	*original pass*
GR	*shot then rewound to frame 1*
INDEX	*to return to the Start position*
XO,8,4,4	*strobe out 8 frames every 4th position*
GO,101	*shoot the strobe* ***will finish at frame 109 but the System Current Frame will be 101***

The System enters the fades for strobes at System current frame 20,000 and manipulates the current frame as necessary to get the desired results. A move entered at frame 20,000 in a conventional manner will be executed on every strobe fade. Therefore if counts above 20,000 are used for the conventional move description this must be taken into account.

EXAMPLE: *We wish to do a conventional move from 1 to 24 with an 8 frame strobe out at each position of the conventional move and with each strobe position moving up as well as fading out.*

OPEN	*Open shutter*
WEST,250,1,24,1,1	*Conventional pass*
CU,1	
ALL	*Shoot first full intensity and strobe*
UP,500,20000,20007,1,1	T*his will Zoom up 500 on each strobe pass*
XO,8	
GO,24	

A strobe may be suspended for revision but cannot be restarted except from the beginning, <u>and</u> you must use OM,X to turn off the special routine flags and then retype the strobe description, CUrrent frame and target frame.

NOTE: (1) It is necessary that before running a strobe that you either OPen or BLack the shutter to initialize it in the proper position for the strobe.

(2) Obviously, fades or dissolves cannot be used in a conventional manner within a strobe routine.

ALL COMMAND

Section 3-31

Format: **ALl**

When a strobe is being executed no full intensity frame is photographed. That is, the shutter fades before an exposure is made.

This is done to make it easier to, for example, make a solid letter move with an outline letter strobe.

The ALl command causes a frame to be exposed **before** the fade starts, shooting ALL intensities of the fade.

ALl may be entered any time before a terminator.

ALl has no effect unless the strobe routines are being used.

COLOR WHEEL OPERATION AND NEXT COMMAND

(See also CINEx)

Section 3-32

The color wheel is a precisely cut aperture disc which holds up to 20 colored gels or other filters or objects. The color wheel is rotated by the computer in conjunction with other moves on a single color-step basis.

Operation:

The color wheel is attached to the drive system in front of the camera lens. The window of required color is centered under the lens and the disc is tightened in position. If the colors are to be moved in a random mode the NExt command is entered and the number of windows required to move ahead of the present window position.

NEXT Color

Format: **Next{,number of window positions to move}**

If no entry follows Next, the wheel is moved 1 window position

EXAMPLE: *Color #1 is under the lens and #5 is required.*

Next,4 *is entered and the wheel moves to #5 position. That is, 4 windows are skipped.*

EXAMPLE: *The color wheel may be used with XI,XO,XB, MUlti, and SMulti and is the last entry on a special routine's function line. (Spacing added only for reading clarity DO NOT TYPE SPACES IN A COMMAND)*

```
e.g.  using MULTI or SMulti
        MUTLI,        2,       5,       (4)
        SMulti,                         (COLOR WHEEL)

e.g.  using STROBE
        XO{XI XB},    1,       1,       1,       3
                                                 COLOR WHEEL
```

In these routines when the number of colors (Wheel Windows) entered have been used, the System resets the wheel to the first color and restarts. Therefore, if 5 is entered only the first 5 colors (windows) on the wheel are used and they are repeated if necessary.

CINEX

Section 3-33

Format: **CINEx** (**Note! 4 letters are required**)

The CINEx Routine is used to CINEx or color/density test a scene. The CINEx Routine requires the use of the color wheel.

When "CINEx" is typed the System terminates the entry of data and shoots 1 frame in each of the 20 positions of the color wheel. It then caps the shutter and runs at advance speed for 3 frames, types "NEXT" and waits with the hand control functions active. If "Continue" is pressed, another CINEx is done. If "REVISE" is pressed the Cinexer stops and the System goes back to the terminal for commands. Typically, an exposure adjustment is made between cinex runs in order to produce a wide range of potential corrections.

Pressing "REVISE" is the only way to stop the CINEx.

Normally, for a scene CINEx, the gels to place in the wheel in clockwise order are:

Window #1 - Blank Window
Window #2 - 10 Red
Window #3 - 20 Red
Window #4 - 30 Red
Window #5 - 10 Green
Window #6 - 20 Green
Window #7 - 30 Green
Window #8 - 10 Blue
Window #9 - 20 Blue
Window #10 - 30 Blue
Window #11 - 10 Cyan
Window #12 - 20 Cyan
Window #13 - 30 Cyan
Window #14 - 10 Magenta
Window #15 - 20 Magenta
Window #16 - 30 Magenta
Window #17 - 10 Yellow
Window #18 - 20 Yellow
Window #19 - 30 Yellow
Window #20 - Blank Window

Cinexing is principally used to determine the color correction required on a product setup, or a transparency. And each successive cinex run is made at a different exposure.

HIT

Section 3-34

Format: **HIT, # of Frames**

HIt will cause the computer to hit (expose) the number of frames in each position where one1 frame would normally be exposed. **This command works in <u>all modes</u>** (MULTI, WIND, LEADER, etc.). **The System Current Frame will advance as though HIt was set to 1**.

EXAMPLE:

HIT,2

CU,1

SH,11

END

This will cause the computer to shoot 20 frames (2 in each position). The CUrrent frame at the end of the run will be 11, and the physical camera frame counter will read 21.

HIt can be removed (Set to 1) by typing:

HIt,0 -or- HIt,1 -or- **any time ZREO is used.**

HIt may be entered or changed any time before a Terminator is entered.

CYCLE

Section 3-35

Format: **CYcle, X, Y, Z**

X = Number of cels or pieces of art

Y = Number of exposure per piece of art

Z = Number of repeats of cels or pieces of art and exposures

The CYcle routine allows the operator to shoot a large number of repeating sequential pieces of artwork in a very efficient manner. CYcle is particularly useful when groups of cells are used to depict an action that repeats a number of times. Such as a character walking where only a few cells are needed to create the illusion but the scene may be much longer requiring that the drawings be repeated several times over.

EXAMPLE: *We have five cels and desire three exposures per cel and thirty repeats, we would write:*

CYcle,5,3,30

It would also be necessary to enter frame data and any desired camera/table/peg moves needed. The target frame entered is accessed by the CYcle routine, the PLatten lifting and current frames are accessed but **the program will execute the sequence until the entire CYcle is completed regardless of the target frame given.** Therefore, a sequence may contain 50 frames but the target frame can be set for any number of frames greater than the CUrrent. The computer will finish only when the data given in the CYcle line are satisfied. All movements, et cetera, are accessed by the CYcle routine in exactly the same manner as in the "Standard" mode. Upon typing the terminator statement, the computer will type READY, (or begin to run); the operator then places the first piece of art on the table (and presses CONTINUE if necessary). The computer will shoot the first piece of art along with any movements necessary in each frame position indicated by the call. In our example, it would shoot the first piece of art at frame 1, 16, 31, et cetera, making all movements indicated between so that artwork was in the proper position at each frame. The shutter would be capped for frames of film which do not contain that artwork. At the end of the frames to be exposed with cel #1 the computer will reverse all directions and reverse the camera motor, cap the shutter, advance back to the starting position, skip over the first frame exposed and type the following message:

NEXT CEL PLEASE MANUAL ACTION REQUIRED CURRENT FRAME XXX

(xxx=frame # showing on camera frame counter)

READY!

The operator then removes the first piece of art, replacing it with the second, presses the "CONTINUE" button and repeats the procedure until the message received from the computer is:

Cycle Finished

CURRENT FRAME XXX (= frame # showing on frame counter) DATA?

To revise from a CYcle operation, the operator must type ZERO or OM,X before entering new data. Since the CYcle may not be revised and resumed once started, it must also be re-entered.

It is necessary to type either OPen or BLack before running the CYcle to insure that the shutter is exactly open or closed. This allows the cycle to be used with fades and dissolves, but it is **necessary even if the shutter is not being run**.

MULTI

(See SMulti, PAuse, PRint)

Section 3-36

Format: **MUlti, #of frames of separation, # of repeats required (Less One), # of colors**

This command allows the operation of multiple passes with or without the color wheel without the need for manual data input for every pass.

The entry is entered at the end of a regular move-execute sequence of commands. MULTI is a TERMINATOR and the System will signal READY! when the terminal RETURN(ENTER) key is hit. It is described the following manner:

To use MUlti with a WAit after every run, a PAuse command is necessary. This stops the operation of the move prior to indexing to the next run's starting position and enables manual action if required. PAuse is entered before MUlti. Since **MUlit is a terminator it also requires a valid current and target frame.**

EXAMPLE: *A move which requires 2 frames of separation, repeated 10 times with 4 colors is entered in this manner:*

AFwd,1,50,1,BO *(Any move or combination or no move)*

CU,1

SH,50

MULTI,2,9,4

Please note that the separation may be any number of frames, including 0. It is necessary with all Special Functions to enter either OPEN or BLACK to initialize the shutter before running.

STATIONARY MULTIPLE

(See PRint, MUlti, PAuse)

Section 3-37

Format: **SMult, #of frames of separation, # of repeats required (Less One), # of colors**

SMulti uses the same data entry format as does MUlti. The visual effect on the film is the same whether using MUlti or SMulti. Also the PAuse command may be used in either instance.

The different between MUlti and SMulti lies in the way in which the same screen effect is achieved.

Where MUlti shoots the described move several times moving and indexing each time like a series of automatic "AGains", the SMulti moves through only once, manipulating the film back and forth to achieve the same screen result.

As a rule, if a long distance move is being done in a short number of frames the SMulti will do the job faster. If a great number of frames are run over a relatively short distance, usually the MUlti will complete faster.

It is necessary to either OPen or BLack the shutter prior to running or SMulti.

PRINT

(See PAuse, MUlti)

Section 3-38

Format: **PRint, #of frames of separation, # of repeats required (Less One), # of colors**

The PRint routine is exactly like MUlit with PAuse except that the **motors are NOT returned at the head of each run**.

PRint is normally used when no motion is required.

PRint will prompt the operator with "NEXT" at the end of each pass, and wait for "CONTINUE" to be pressed.

HOLD

Section 3-39

Format: **HOld, # of frames to hold**

The HOld command is designed to run a variable number of frames in position. HOld is always typed with the number of frames to be held (Exposed without advancing the Current Frame). As soon as the command is sent, the computer will type READY! When "Continue" is pushed, the indicated number of frames is run at normal speed and DATA? is printed. . . **NO MOVES ARE EXECUTED and the computer Current frame is NOT changed.**

EXAMPLE:

ZEro

CUrr,10

HOld,100

READY *system responds*

The computer will shoot 100 frames when “Comtinue” is pressed and then stop...the current frame in the computer will still be set at frame #10.

LEADER

Section 3-40

Format: **LEader,y**

y = A number of frames to Leader (run off capped) **NOT a target frame**, LE,1000 will run off 1000 frames, regardless of the system CUrrent frame.

LEader CAps, sets high speed, runs the number of Frames entered, then returns to the terminal for more DATA.

GO THEN REWIND

Section 3-41

Format: **GRewind{,Number of frame to rewind or fast-forward to}**

If a rewind to Frame is not entered, then the System will Rewind to the Current frame which existed at the time the GR command was typed. **When finished the System Current frame is reset to the value entered in the GR command**.

The GO REWIND Terminates like the GO Command, that is, the will begin to run as soon as the GRewind command is typed

EXAMPLES:

(1) Shoot from 1 to 100 then rewind to 50

CU,1 *(Establish current frame if desired)*

SH,100 *(Set target frame for normal shooting)*

GR,50 *(Terminate with 50 as the frame to return to)*

(When the system is the finished CUrrent frame = 50)

(2) Shoot from 1 to 100 then cap and Fast ***Forward*** *to frame number 150*

CU,1

SH,100

GR,150

(When finished CUrrent frame = 150). Frames from 100 to 150 were skipped over at high speed, shutter capped,

(3) Shoot from 1 to 100 then Rewind to 1

CU,10

SH,100

GR *Shoots from 10 to 100 & rewinds to Current Frame 10*

END AND REWIND

(See ENd)

Section 3-42

Format: **ERew{,Number of frame to rewind to}**

End Rewind behaves exactly like GRewind except that it terminates like the ENd statement. That is, it waits before and after Running.

HOOKUPS

Section 3-43

Format: **(2 Letter I.D.), x, y**
x = The Counter Number not to move across (extreme)
y = The Counter to Return to

Hookups are normally used only on Peg track or Rotation Channels. The list of 2 Letter Hookup I.D. entries can be found in Appendix "D" of this manual.

The primary purpose of a hookup is to allow a short piece of artwork to be run as though it were longer. Such a situation exists when, in cartoon animation, a background is drawn, say 24 inches wide with the left side and right side containing the same picture information. A Cycle of cels is used to depict a moving character, but the background is panned by several times to give the illusion of motion. Another use of Hookups is in the photography of long crawl titles, say 10 feet of artwork registered on a 3 foot peg track.

The Entry for the number that the System is not to move across is one extreme of the motion. When a move is run that would cause the System to move across that point, the System automatically moves to the point entered as the return to point and moves the excess distance from that point. It should be noted that the Hookup extreme and Return to points will imply a certain direction of travel. **A hookup will be executed only if the direction of motion is the same as the implied direction <u>and</u> the hookup extreme would be exceeded**.

EXAMPLE:

H3,100,0 *(**H3** is the 2 letter I.D. for Peg track #3)*

This means: *(1) Do Not Exceed counter reading 100 on track 3*
(2) Return Track 3 to 0 if 100 is exceeded
(3) The implied direction is from counter reading 0 towards 100

Please note that the indexers do not access the Hookups. Note also that you can move beyond a hookup point with the Hand Controller if you wish time to adjust things before the Hookup occurs. A PAuse command will cause the Hookup operation to print a message and wait for you to push "Continue" before returning to the Hookups Return point.

To remove a hookup from an axis, enter the 2 letter I.D. and 0, 0 as the extreme and Return to points.

EXAMPLE:

To remove Track 3 Hookup

H3,0,0

PROBLEM: We have an 8 foot piece of artwork to shoot and Track 3. Track 3 can only move 3 feet (36 Inches). The move (the whole 8 feet) is to run from frame 100 to 300.

SOLUTION: The distance between 2 pegs on an ACME or OXBERRY peg track is 4 inches. We will assume that the counters are 1/100th of an inch - therefore 4" = 400 units on the counter. We place the artwork on the first peg of the track and position the track and camera for the head of the shot. We'll say that the counter reading for the Head of the shot is 8500 on the counter.

Then we pan the track as far as we can and still stay in the field and then back it up to the nearest multiple of 400 units from the start, and note the counter reading.
In this case, let's say that is counter reading 1700 or 32 inches of movement. 8500 = minus 1500 so 1500 + 1700 = 3200 = 32 inches
The Hookup extreme then is 1700 and the return to point is 8500.

The entry to the system would be:

H3,1700,8500

The implied direction would be from 8500 towards 1700.

The System will not pass 1700 <u>in that direction</u>. The total distance is 8 feet or 9600 units. The move command for track 3 would be:

3E,9600,200,300,10,0

PAUSE

We will use PAUSE to make the System wait before doing each hookup.

When we RUn - and the System indicates a hookup is about to occur, we take the 8th peg (32 inches from the start) on the artwork, lift the artwork and position it to the 1st peg.
This is repeated each Hookup, eventually photographing the entire piece of artwork

SET LENS FOCAL LENGTH

(See also FLength Exponential Zooms)

Section 3-44

Format: **SLens**

The SL command is used to allow the system to determine the exact focal length of the lenses you are using, and to gain information about the animation device.

Before the SLens command is used, the scale for the zoom motor must have been previously set. See the Direct Scale and Veeder sections for information as to how this is done.

When the SLens command is received, the system will request that you enter the system security code to avoid accidental resetting of lens focal lengths.

The Set Lens routine is interactive and will request information regarding the lens and film aperture being used. The system will then ask that you position the **FILM PLANE** two feet above the table. It is important that this position be as accurate as possible. After positioning to two feet, push 'Continue'. The system will then ask that you position to three feet above the table. Push 'Continue' when you have reached the three foot position. You will then be asked to move to exactly twelve fields. At this point, you should move to a point where twelve inches exactly fits within the **FULL ACADEMY** or full 16 mm aperture. Note that this is beyond some rack-over projector’s aperture. When you have reached this position push 'Continue'. The system will then print the focal length in millimeters for the lens that you are using.

Note that the **two foot and three foot positions only need to be entered the first time you use the SL routine.** On subsequent SL requests - the system only requires the twelve field position.

IDENTIFY LENS

Section 3-45

Format: **FLLO(or FLSH)**

This instruction is used to indicate to the system which lens **F**ocal **L**ength is in use Long (LO) or Short (SH). This focal length is used by the exponential routines and by the **F?** size routines.

IDENTIFY APERTURE

Section 3-46

Format: **AP35 or AP16** (all 4 letters required)

This instruction identifies whether the 16 or 35mm aperture is being used. This information is used by the F? routines for determining field size.

PRINT SIZE AND FIELD SIZE

(See FL and AP)

Section 3-47

Format: **F?**

When the system receives this command, the current size and field size will be displayed. In order that these data are correct, you must have first identified the lens (see FL) and aperture (see AP) sizes.

MATCH MOVE AT ANOTHER FIELD

(See MC)

Section 3-48

Format: **MAtch, Frame Number, Zoom Height**

The Match move routine is used to duplicate the apparent speed of a move at a different field size exactly. The Set Lens, aperture and focal length must have been set previous to using this command. The frame number entered indicates the first frame of the move to match. The zoom height entered is the zoom counter reading of the **original** zoom at the frame entered as the first frame of the move to match.

In order for the system to match a move at a given frame, it is necessary that the System CUrrent frame not be contained within the move being matched. For example, an original move from 1 to 100 then the CUrrent frame set to 50 and the SHoot target 100 in the match move will result in the system issuing an error, as this would cause the compounding of the original move (move at frame 51 plus match at 51).

EXAMPLE: *We have previously entered (**but not ZEroed**) a move starting at frame 20 running to frame 120(100 frames of motion). The original move was shot starting at Zoom Counter 5500. Now we wish a speed matched move starting at Zoom Counter 4500.*

TR,4500	*Move to 4500 (any time before running)*
MA,20,5500	*Setup frame and height*
CU,1000	*Make sure Current is greater than 120 (end of original move)*
SH,1100	*The original move was 100 frames*
GO	

At this point, the system will determine the present zoom height and execute the move at frame 20, proportioned to the current height. IF any other moves were entered from 1000 - 1100, they would also be executed, but will not be proportioned, since it is implied that they are at the current size.

When the move finishes, the MAtch mode is automatically ended.

If REVISE or EMERGENCY STOP is used, remove this special operation with the OM,X command.

Note: MAtch cannot be used with STrobe, SCan or TUbe, but can be used to FIgure or MFigure data for SCan.

MATCH CONTINUE

(See MAtch)

Section 3-49

Format: **MCont, Frame number of move being matched**

Match Continue (MC) is used when you wish to continue a MAtch move. MC leaves an indicator that the MAtch Mode is on and **must be removed by OM,X (or ZEro)** when you wish to stop the MAtch Mode.

In order to use MC, you must first use the MAtch mode to setup the original zoom counter reading.

EXAMPLE:

MA,1,5000	*Match a move at 5000 current frame 1 in original*
CU,1000	*Setup a 'Suitable' Current Frame*
GO,1001	*Shoot one frame matched to the current size*

At this point, one frame has been matched, and the Match mode is switched off by the system.

MC,2	*Turn on Match at correct next frame; size of this frame is automatically calculated*
GO,1002	*Shoot one more frame matched*
GO,1100	*Shoot 98 more frames matched*
OM,X	*OFF Match*

CRt Mode

(See TTy)

Section 4-1

Format: **CRt**

The CRt mode is used with systems supplied with and ISC (Intelligent Systems Corporation) color terminal. The CRt will operate **ONLY** with this color terminal. When the system is in the CRt mode the frame counts and positions are continuously updated on a frame by frame basis and displayed in white letters on a red background. This information is placed in the upper right hand corner of the screen.

To return to a normal mode of operation type TTy.

DO NOT USE CRt UNLESS YOU ARE USING AN ISC TERMINAL!

*(*Hand written note * If TTy is not operational go to HALT, M-4062 (11-5-4-1) LIGHTS, THEN STORE, T-0 SHOULD SHOULD SHOW, CLEAR DISP, STORE, RUN)*

Editor's note on handwritten note: The instructions written above in blue may not be valid, they are definitely confusing. Whether or not they will be valid depends upon whether or not the system software was upgraded after the note was written, so **tread lightly**.

TELETYPE (NORMAL) MODE

(See CRt)

Section 4-2

Format: **TTy**

The TTy mode is the system's normal mode of operation and may be used with any Cinetron supplied CRT, Terminal, Personal Computer or Teletype, including the ISC Color Terminal.

The TTy command is implied and TTy need only be used to switch back to the system's normal TTy mode from the CRt mode when using an ISC Color Terminal.

No harm will come from typing TTy while in the normal TTY Mode.

DRAW

Section 4-3

Format: **DRaw**

The DRaw command causes the system to draw on the screen of the Intelligent Systems Color CRT. The representation drawn is that of any movements the system has instead of moving the motors. The fields drawn are calculated by the system as if 16mm film and a 55mm lens were being used. The system will calculate the positions of a motion and then draw on the screen of the color CRT, the field size and position from the screen center that the motion indicates.

DRaw will work only with the Intelligent Systems Color CRT.

To REMOVE Draw type: EXpose or ZEro

NOTE: ONLY ZOOM, NORTH/SOUTH and EAST/WEST positions are drawn.

AUDITION

(See FAdes and DIssolves, EXpose)

Section 4-4

Format: **AUdition**

When AUdition is typed the system will set a condition which prevents exposing film.

When movements are run the system will hesitate briefly in the position where a frame would normally be exposed, then run to the next position without exposing film.

AUdition allows a method of previewing moves without shooting.

THE SYSTEM CURRENT FRAME IS ADVANCED WITH EACH FRAME AUDITIONED

To remove AUdition type EXpose or ZEro.

EXAMPLE: *We wish to preview a programmed move before shooting, then, if it's OK we wish to shoot it.*

ZERO	*(Not necesary)*
LOCK	*(Remember this position)*
CU,1	
EAST,100,1,24,6,7	*(East motion described)*
AUdition	*(Turn on AUdition)*
GO,24	*(Target frame = 24)*

The System will run through the move, pausing briefly in each frame position - no film is shot.

To shoot this move:

INdex	*(Return to LOck position which is the head of the shot)*
EXpose	*(Turn off AUdition, See next section)*
AGain	*(Run last move again)*
READY!	*(System responds)*

When *"Continue" is pressed now the system will shoot whatever was just AUditioned.*

Note that AGain always resets the Current and Target frames.

EXPOSE

(See AUdition, DRaw, WRite)

Section 4-5

Format: **EXpose**

Expose removes the AUdition, DRaw, and WRite modes and sets the system to expose film.

It is not necessary to use EXpose unless DRaw, AUdition, or WRite has been used and ZEro has not been typed since one of the listed commands was used. EXpose is the default system condition.

EXAMPLE #1:

EXpose	*(Remove DRaw, AUdition & WRite)*
ZEro	*(Also removes DRaw, AUdition & WRite)*

EXAMPLE #2:

AUdition	*(Turn on AUdition)*
ZEro	
EXpose	*(Not needed but no harm is done)*

PLATEN CONTROL

(See OMit)

Section 4-6

Format: **PLaten**

The PLaten operation is used to cause the system to raise and lower an automatic cel platen to prevent tearing cels. The PLaten command may be used even if an automatic platen assembly is not installed on your stand, as the time delays set to allow an automatic platen to operate are long enough for you to manually lift between moves IF you are paying attention.

To remove the PLaten command type: ZEro or OMP

PAUSE

(See Strobes, Hookups, MUlti, SMulti)

Section 4-7

Format: **PAuse**

The PAuse command will cause certain automatic routines to wait until "CONTINUE" is pressed before performing certain actions. You should refer to the explanations of the specific routines to determine the exact operation of PAuse on a given operation.

PAuse has no effect on normal shooting, only on certain special routines.

To remove PAuse type ZEro or OMX.

ACCURIZE

Section 4-8

Format: **ACcurize**

ACcurize is used to cause all motors to be left in a direction which the system considers to be accurate.

The System considers clockwise rotation of a motor to be its accurate direction. If ACcurize was typed, and if the system moves a motor in a counterclockwise direction, it will continue to move in that direction past the indicated position, then reverse and move back to the indicated position. This insures that slack in chains, gears and lead screws is nullified.

ACcurize is not required in all but the most critical situations or where the mechanics of the machine are known to be in terrible condition.

To remove ACcurize type ZEro.

CAP

(See UNcap)

Section 4-9

Format: **CAp**

The CAp command causes the System to close the capping shutter as soon as the command is sent.

Normally the capping shutter is controlled automatically by the Special Routines such as SCan, CYcle, MUlti, etc.

The CAp command is normally of little use unless the System is being run from an external computer or a Transfer File.

UNCAP

(See CAp)

Section 4-10

Format: **UNcap**

The UNcap Command causes the Capping Shutter to open as soon as the command is sent.

UNcap is the opposite of CAp, and its functional use the same as CAp, except that it opens instead of closing the capping shutter.

MOTOR

Section 4-11

Format: **MOtor**

The MOtor command, when typed to the System, causes the Stop Motion Camera Motor to reverse directions. If the camera is running forward when MOtor command is read by the System it will, at once, set the motor to run in reverse, it in reverse the motor is set to run forward.

Like CAp and UNcap, MOtor is normally used only when running from an external computer, or in a transfer file.

CONVERT FRAMES TO SECONDS

Section 4-12

Format: **?S, Number of Frames**

Converts frames to seconds.

EXAMPLE:

?S, 121

The computer will type: **5 seconds + 001 frames**

CONVERT SECONDS TO FRAMES

Section 4-13

Format: **?F, Number of Seconds**

Converts seconds to frames.

EXAMPLE:

?F, 10

The computer will type: **240 frames**

POSITIVE

(Convert Negative Veeder Root Number)

Section 4-14

Format: **PO, Counter Number (from 5001 to 9999)**

Converts the Negative Veeder Number to Positive

EXAMPLE:

PO,9900 is typed

The computer will type **100**.

BASIC

Section 4-15

Format: **BASIc** **Note that 4 letters are required**

The BASIc command is used when an external BASIC language programmed computer is communicating with the system.

BASIc is of no value in any other mode of operation. When BASIc is received by the System it will, at once, look at the hand control switches and wait for "CONTINUE" to be pressed. When "CONTINUE" is pressed the System will start looking at the **terminal** and disregard everything it sees until "**OUT**" is received. Once "**OUT**" is read on the terminal line the System returns to a normal mode of operation (**i.e. to remove BASIc type "OUT"**.) This sequence of operation allows the external computer to be connected in parallel with the System.

TRANSFER FILES

Section 4-16

General Information

The transfer file consists of 59 lines of information which you may "file away" in the System Computer's memory. Alternately **SFile** may be used to enter a group of lines. **An up arrow (^) as the first character typed is used to identify the remainder of a line being typed as a line for the System to place in the file.** The System "Strips Off" the up arrow and places the rest of the line in the transfer file area.

There is a command, **ZF**, which clears the file and sets the System to write your next entry to the file on the first line.

There is a command, **WF,** which causes the system to write all or part of the information in the file so that you may see what's there.

There are 9 file variables, Global Values, which you may use to manipulate the operation of the file or substitute for System values. **These values are Global in scope, that is, they retain their values until they are explicitly changed and they are not exclusive to the file in which they are created**. The values may be modified, cleared and/or inspected from different transfer files and by the operator.

Information in the file consists of comments, normal system commands, and commands which are used in the operation of the file itself.

All transfer file commands, which direct control of the file itself, start with a colon ':'. The one exception is the command to transfer out of the file '//'.

If the System is reading data from the transfer file and the command does not start with a colon, then it is assumed to be a normal system command, and is sent to the System exactly as it is saved in the file.

A command ‘TF’, when entered from the terminal, causes the system to "Transfer" to the file at a specified line number.

While in this mode, the System will not display commands which effect how the file is to operate (those starting with a colon :). Normal system commands found in the file are printed or "echoed" to the terminal so that the operator can follow what is going on unless the blind transfer ‘BT’ mode is in effect.

Most errors, when detected, will cause the System exit out of the transfer mode. If "Revise" or "Emergency Stop" is pressed the System will exit the transfer mode and return to the Terminal.

PLEASE NOTE! The number or pound sign ‘#’ cannot be used within a transfer file.

FILE COMMANDS FROM SYSTEM MODE

Section 4-17

Format: **WFile (,x,y)**
x = First line to write
y = Last line to write

The Write File command causes the System to respond:

Transfer Command File

Next line to enter = xx (xx = LIne #)

This message is followed by a list of all 59 lines in the file. Each line is numbered. IF nothing was entered on a given line since the last ZFile (Zero File) command, then a double slash or transfer out of file command will be on the line.

The "next line to enter" refers to the next line which will be written into when the System receives an up arrow '^' as the first character of a data entry or if the SF command is used.

If a number is entered as the first line to write, then the listing will start at that line, and continue to the last line to write which was entered, or to the last line of the file if no last line number was entered.

ZERO FILE

Section 4-18

Format: **ZFile**

When the System sees the ZFile command several things happen:

1. The entire system is filled with transfer out commands (//).
2. The file is set to enter the next data on line #1.
3. The System types:

   Transfer File Cleared!

SET WRITE LINE NUMBER

Section 4-19

Format: **SWrite, Line Number**

The set Write command sets the file to write into the file at the specified line number.

The typical use of the SWrite command is to facilitate editing data in an existing file.

The transfer file routines automatically set to write on the next line after data is filed.

EXAMPLE:

We wish to change line 6 in the transfer file. The WFile command indicated that line 30 would be the next line to write.

Assuming this is the data in the file:

(Line #)3	**CU,1**
4	**SH,48**
5	**GR**
6	***Put on Art63**
7	**East,100,2,24,3,IN**
8	**WAit**
.	*other data entered*
29	**GR,77**

Then later we enter:

SW,6	*(Set to write on line 6)*
^*Put on Art62	*(Then put in the corrected data*
SW,30	*and reset to write on line 30, otherwise the next line starting with (^)would be written at line 7.*

Now the data will read as follows:

Line 3	CU,1
4	SH,48
5	GR
6	*Put on ARt #62
7	East,100,2,24,3,IN

.

.

.

.

.

ZERO GLOBAL VALUES

Section 4-20

Format: **ZGlobal**

The ZGlobal instruction sets **all nine** Global values in the transfer file equal to zero ‘0’.

ENTER DATA TO THE FILE

Section 4-21

Format: **^ (as the first character of the line)**

In order to place (enter) a line in the transfer file, the first character typed must be an up arrow '^' unless SF command is used (See Section 4-41).

The validity of the instruction is not checked when entered. Validity is checked only when the instruction is read under the control of the file (See Verify and TFile). At that time it will be checked to see whether or not it is legal.

SPECIAL NOTE REGARDING COMMENTS:

It is often desirable to include comments within the file in order to notify the operator of various things.

The Cinetron System **totally ignores** spaces sent via the Terminal; therefore, **it is not possible to place a space character in file.** This presents a problem in that all spaces in a comment line will be "stripped off" by the System causing the remaining character to be "packed" against each other.

To avoid this, use a dash, period or other printing character to separate words in a comment line.

EXAMPLE:

*** This is a comment**	Your entry
**Thisisacomment*	*Would look like this when "sent" from the file to the Terminal.*

--However--

*** - This - is - a - comment**

Would Appear as

***-This-is-a-comment**	*When sent To the user's terminal*

TRANSFER TO FILE

Section 4-22

Format: **TFile, line number**

The TFile instruction transfers control of the System to the file beginning at the designated line number. The System will be under the control of the transfer file and unless REVISE, EMERGENCY STOP, or an ERROR occurs, the file will retain control until a transfer out instruction '//' is found in the file. The System begins reading the file at the designated line number and executes each instruction according to its contents.

FILE MANAGEMENT COMMANDS

Section 4-23

General Information

File management commands are those which start with a colon ':' and control the operation of the file itself. **File management commands have no meaning in normal system operation and will be considered as errors**.

The transfer file routines read these commands and decide what to do, and what to "send" to the System.

There are nine "Global values" which reside within the transfer file. These 'Globals' are numbered 1 through 9. Each Global has a value associated with it which you are able to set and modify. Global values are used as counters and as values to substitute in normal system commands. **Please note that the Global number is the name of the value, not the value of the Global number**. For example, Global 5 may have any value from -32767 to +32767 assigned to it (See Set Global Value).

In addition to the transfer file itself, there may also be up to 20 copies of data which were stored in memory from the transfer file itself (copies of transfer files, see NAme).

A Global value will not be changed unless a file instruction changes it. Therefore, these values may be accessed by other files loaded to the transfer file (See GEt).

JUMP TO LINE NUMBER IN FILE

Section 4-24

Format: **:J,x (line number)**

The Jump Command causes the file to jump to the indicated line in the file and read it next. **If no number is entered, then the next line is skipped.**

The Jump Command is useful to modify the normal sequence execution of the file.

EXAMPLE: *We wish the System to read lines 1 through 10 of the file then go to line 50 then at 55 go back to line 11.*

The lines would look like this:

Line 9 **(some command)**

10 **:J,50**	*Command to jump to line 50*
11 **(command)**	
.	
15 **:J**	*Will cause the system to skip line 16 Since no number was entered*
.	
.	
50 **(another command)**	
.	
.	
.	
.	
.	
55 :J,11	*Command to go back to line 11*

TRANSFER EXIT

Transfer to Normal Mode

Section 4-25

Format: //

If the transfer file reads a double slash (//) as the first 2 characters stored on a file line, then the transfer file will print:

TRANSFER EXIT LINE # (line number)

and return to the terminal to read data in the "Normal System" mode.

SUBSTITUTE GLOBAL VALUE FOR SYSTEM CURRENT FRAME

Section 4-26

Format: **:Cx**

x = A Global value number from 1 to 9

When this instruction is read, the File will substitute the value of the entered Global into the System CUrrent frame.

EXAMPLE: Global #6 has a value of 120 associated with it.

:C6

Will cause the System CUrrent frame to be set to120.

SUBTITUTE GLOBAL VALUE FOR SYSTEM TARGET FRAME

Section 4-27

Format: **:Tx**

x= A Global number from 1 to 9

The :T command will cause the File to substitute the <u>value</u> of the Global entered into the System SHoot target frame.

EXAMPLE: *If Global #3 has a value of 14 <u>associated with</u> it*

:T3

will cause the System Shoot target frame to be set to 14

- or -

*You might say **<u>in this particular case</u>** that*

:T3 *is equivalent to SH,14*

But if GLobal #3 was 24 then

:T3 *would be equivalent to SH,24*

SET GLOBAL VALUE

Section 4-28

Format: **:Sx,y**

x= A Global number 1 to 9

y = The **positive** (0 to 32767) value to associate or "put in" the named Global

The Set Global instruction is used to initialize or, optionally, to set the value of the named Global.

EXAMPLE: *We want Global #4 to equal the value of 247*

:S4,247

NOTE: Although you are only able to set a Global number to a positive value with this command , the system itself is capable of assigning a negative value as a result of executing other Global or system commands. See :+ and :- commands, for example.

ADD TO GLOBAL VALUE

Section 4-29

Format: **:+x,y**

x= A Global number from 1 to 9

y = The **amount** (0 to 32767) to **add to** the existing value of the Global

The Add to Global command allows the operator to increase the current value of a Global by the indicated amount.

EXAMPLE: *If the Global #6 equals the value 35*

:+6,3 *Will add 3 to 35 so the value of Global #6 would then the 38.*

If Global #6 has a value of -10

:+6,3 *Would cause Global #6 to have the value of -7 (-10+3=-7)*

SUBTRACT FROM GLOBAL VALUE

Section 4-30

Format: **:-x,y**

x= Global number from 1 to 9

y = The amount to subtract from the value of the named Global

The Subtract from Global command is used to decrease the value of the named Global.

EXAMPLES:

(1) *If Global number #8 has the value of 14 then:*

:-8,3 *Would cause Global #8 to have 11 as its value.*

(2) *If Global number #8 had a value of -10 then:*

:-8,3 *Would cause Global #8 to have the value -13.*

NOTE: The : - command can be used to set a Global to a negative value.

EXAMPLE: *We wish to set Global #4 to -200*

:S4,0 *First set to 0*

:-4,200 *Then subtract 200*

Now Global #4 = -200

SKIP IF GLOBAL EQUAL TO ZERO

Section 4-31

Format: **:Zx**

x= Global number from 1 to 9

This instruction inspects the value of the Global entered. If its value is 0 then the next line in the file is skipped.

EXAMPLE: *We want to avoid typing ZEro to the System except on the first pass. We will use Global #1 to count the number of passes so if Global #1 is equal to 0 then it's the first pass.*

Line #1	**:S1,0**	*Set Global #1 to 0*
#2	***This is the Beginning**	*Comment*
#3	**:Z1**	*Skip if Global #1 = 0*
#4	**ZEro**	*-Send ZEro to System*
#5	**:+1,1**	*Add 1 to Global #1*
:		
:		*(other commands)*
:		
#24	**:J,3**	*Jump to line 3*

In this example the file can run and will execute the ZEro at line 4 only the first time through since the instruction in line 5 will cause Global #1 to be set to 1 on the first time through, 2 on the 2nd, and so forth.

Obviously, if Global #1 is only tested on line 3 and not used otherwise this could be more efficiently written by just making the instruction at line #24 jump to line 5 or 6. Presumably, in the example, Global #1 would have other duties.

TEST GLOBAL FOR EQUIVALENCE TO A VALUE

Section 4-32

Format: **:=x,y**

x= Global number from 1 to 9

y = Value of Test

The Equivalence test causes the transfer file to skip the next line if the value of the Global is equal to the value of the test. Otherwise, the next line is executed.

EXAMPLE: *We want to do a set of lines in the file 25 times, then print a finished message and transfer out.*

Line #1	:**S2,0**	*Set Global #2 = 0*
#2	**.**	*commands*
#3	**.**	*to*
#4	**.**	*be repeated*
#5	**:+2,1**	*Add 1 to value of Global #2*
#6	**:=2,25**	*See if = to 25 (25 times through)*
#7	**:J,2**	*If not, jump to line 2*
#8	***FINISHED**	*If equal line 7 is skipped*
#9	**//**	*Transfer file exit*

SUBTITUTE GLOBALS IN SYSTEM COMMAND

Section 4-33

Format: **:x,nnn,xxxxxxxxxxxxx**

nnn =A string of letters or numbers (**cannot contain a comma**) which represent what is to be sent to the System as a normal command.

xxxxxxxxxxxxx= A series of Global numbers from 1 to 9 which are in the order that they are to be sent to the System.

The Substitute command is used to send a command to the System along with the values of various Globals. The Global values may be sent in any order or may be repeated on the line if needed.

When this command is read in the file, the string of data enclosed by commas is sent to the system exactly as written. Then the values of the Globals are sent in the order that the Global numbers are written.

The Globals are separated from each other by a slash '/' (which is a valid data separator). **Note, however, that the two commas are not sent**. Therefore, normally the last character in the "string" will be either a period or slash to separate the string from the Global values in order to fit the normal system instruction formats.

When decoded, the command is typed on the terminal and "sent" to the System.

EXAMPLE: *Let's say that the Globals have the following values:*

Global #1 = 1000

Global #2 = 200

Global #3 = 30

Global #4 = 400

Global #5 = 501

Global #6 = 6096

Global #7 = 70

Global #8 = 80

Global #9 = 90

if we want to send a TRack command to the System to TRack (to), 1000,400,200, 501

:X1, TR/, 1425	*Would be the command*
TR/1000/400/200/501	*Would be sent to the System*

Using the above values for the Globals:

:X, *GLOBALS =, 123456789	*This command*
***GLOBALS = 1000/200/30/400/501/6096/70/80/90**	*Will send this comment*

SAMPLE TRANSFER FILE

Section 4-34

We want to have the operator run to the end of a series of moves. Each move will start at a different position and will be 24 frames long, and there will be 50 of these. Before each move the operator will run to the head of each move manually.

Line #		
1	**ZG**	*(Clear Globals to 0)*
2	***SAMPLE - TRANSFER - FILE**	*(comment)*
3	***RUN - TO - THE - END - OF – ALL-THE - SCENES - PUSH - CONTINUE**	*(comment)*
4	**WAIT**	*(Wait for operator)*
5	**LOCK**	*(Memorize end position)*
6	***RUN - TO - THE - HEAD - OF - THE - SCENE**	
7	**WAIT**	*(Wait for operator to press continue)*
8	**ZEro**	*Now clear old moves*
9	**CU,1**	*New move*
10	**ARev,1,25,3,4**	*Instructions*
11	**GO,25**	*Shoot move*
12	**:+1,1**	*Add 1 to Global 1*
13	*	
14	**:X,*END-OF-PASS,1**	*Print comment & pass # just done*
15	**: = 1,50**	*Check if 50 done*
16	**:J,6**	*No so get new head position*
17	***END OF JOB**	*Yes tell him*
18	**//**	*And transfer out of file*

When we type **TF,1** the System will start reading the file.

First the operator will be given the chance to position to the end of the move and press "Continue" when he gets there. Then he moves to the head of the move and presses "Continue".

The System enters the move, runs it, tells the operator which run was finished and goes back to allow the operator to select a new beginning position. When that move is done, this continues until 50 moves are completed. When the file loop is finished, the System will go to a "Normal" mode. If the film were rewound at the end of each move an under lit piece of art would create a final in which 50 objects converged to a single point.

INPUT VARIABLE FROM TERMINAL TO GLOBAL

Section 4-35

Format: **:Vx**

x= A Global number from 1 to 9

This instruction allows the operator to define the value of a Global from the Terminal. The major use of the :Vx command is to allow for writing a general purpose file and being able to redefine the variables without rewriting the file.

EXAMPLE: *We want to type the line * command as many times as the operator desires. We will use Global #5 to decide how many times to write the line.*

Line 1	*** HOW-MANY-TIMES-?**	*Ask the operator*
2	**:V5**	*Input the # from the operator*
3	**:Z5**	*Skip if 0*
4	**:J**	*Not 0 So Skip*
5	**//**	*If 0 transfer out of file*
6	***COMMENT**	*Print the line*
7	**:-5,1**	*Subtract 1 from Global*
8	**:J,3**	*Jump back and test for 0*

SAVING, PURGING, LOADING TRANSFER FILES

(See also Section 4-40)

Section 4-36

General Information

The System is capable of saving the data from up to 20 different transfer files.

To save the data in a transfer file, use the NAme instruction (Section 4-37).

To wipe out one of the NAmed files, use the PUrge instruction (Section 4-38).

To load a file into the transfer file, use the GEt instruciton (Section 4-39).

To list all the NAmed files saved in memory and their individual security codes; use the WNAme instruction (Section 4-40).

NOTE: All of these are System commands and may be used within a file or from the keyboard.

PLEASE NOTE! Global values created or set in one file may be accessed by another file and modified if appropriate.

SAVING TRANSFER FILES

Section 4-37

Format: **NAme, (up to 6 character name){; up to 2 character security code}**

The data currently in a transfer file may be saved for future use with this instruction. Each time a file is NAmed, the System adds the file name to a list which is used to identify the file and tell the System where to find it. This entry is called an **ID segment**. IF an attempt to save more than 20 files is made, an error (**error #45**) will occur. This is because there are only 20 ID segments available (20 file names). If this error (ignored #45) occurs, then a file <u>must be PUrged</u> (Section 4-38) in order to release an ID segment for use.

Other errors which may occur are:

Ignored #46 . . . indicates that the name you are assigning to a file has already been used, and that the file still exists. To correct this, either PUrge the old file or select a new name for the new one.

Ignored #47 . . . FILE AREA FILLED indicates that the entire space to save the files is now filled, and that, even though 20 files have not been named, that the storage space for the files is used up. This will usually indicate that several of the files previously saved is abnormally large. The only correct action is to PUrge enough area to save the new file.

Ignored #48 . . . Indicates that an ID segment was available and that some storage space was available, but not enough to save the current file. To correct this, either reduce the current file to the minimum (eliminate unneeded instructions), or PUrge an old file.

The file name may be any combination of up to 6 alpha-numeric characters except the SEMICOLON.

The security code may be any combination of 1 or 2 alpha-numeric characters and is <u>separated from the name by a semicolon</u>.

It is not necessary to enter a security code but it is recommended to prevent accidental PUrging of a file.

If more than 6 consecutive characters are entered for the name then **IGNORED #5** (too much data) will result. <u>If more than 2 characters are entered for the security code, then only the first 2 will be used</u>. The security for a file need not be unique.

When a file is saved by the System, the message: **FILE CREATED!** will be displayed by the System, and the file placed in memory along with an ID segment. **The file saved will still be in the transfer file; it is not altered by saving.**

EXAMPLES:

NAme, MYFILE;ME	*Creates a file called MYFILE with security code = ME*
NAme, AFILE21;21	*ERROR..more than 6 characters in the name*
NAme, 1;21	*OK..file name is 1 - security is 21*
NAme, AFILE2	*OK..creates a file named AFILE2 with no security.*
NAme, AFILE2;AFILE	*OK..creates a file named AFILE2 with security of AF*
NAme	*ERROR..no name entered*

When the file name and security is used in the GEt and PUrge instructions it must match exactly the name & security as NAmed.

PURGING A SAVED FILE

(See Section 4-37)

Section 4-38

Format: **PUrge, (6 character name){; 2 character security code}**

The PUrge instruction eliminates the named file and releases its ID segment for reuse. The file in the transfer file is not affected.

The file name and security must match exactly or an error will result and nothing will be purged.

If a file was NAmed without security, it may be PUrged without entering security.

It is not possible to recover a purged file.

When a file is purged the System will respond: **FILE PURGED!**

Purging a file has no other effect on the transfer file area (See ZF).

LOADING A FILE TO THE TRANSFER FILE

(See Section 4-36)

Section 4-39

Format: **GEt, (6 character name)**

A Security code for the file is not required. The GEt instruction loads the file identified to the transfer file. The transfer file is set to read at line one, unless the GEt instruction is proceeded by a SUb instruction. **The current data in the transfer file is overlaid by the GEt file command and is therefore lost** (it can be saved by using the NAme instruction first). If this instruction is used within a transfer file, then the FILE you GEt will continue executing at line 1, or the line indicated by a SUb instruction.

When a file is loaded, the System responds**: FILE LOADED!**

EXAMPLE: *There are 2 files named FILE1 and FILE2. The file currently in the transfer file will load FILE1. It will run and then load FILE2 which will run and then transfer out.*

Data in current transfer file:

GEt,FILE1

FILE LOADED *(System loads File1 and continues to run FILE1at line 1)*

TF,1 *Transfer to file @ Line 1*

Data in File 1:

Line 1 ***FILE 1 RUNNING NOW**

Line 2 **:X,AF121121,34** *Various*

.

. *Instructions*

GEt, FILE2 *File 2 will run automatically from line 1*

FILE LOADED! System response File2 is now running

Data in FILE 2:

Line 1 ***FILE2 RUNNING**	
Line 2 . UP,2231,2,276,18,22	*Various*
.	*Instructions*
//	*Transfer out*

NOTE: GEtting a file will not cause the file to execute unless the instruction is contained within a transfer file which is already executing.

DETERMINING FILE NAMES

(See Section 4-36)

Section 4-40

Format: **WName**

This instruction causes the System to list the names and security codes of all the transfer files in memory.

ALTERNATE METHOD OF ENTERING DATA TO A TRANSFER FILE

Save Multiple Lines in Transfer File

Section 4-41

Format: **SFile**

The Save File instruction causes the System to transmit data entered on the keyboard to the transfer file. <u>This instruction also "turns off" the transfer file if it is running</u>. Therefore, **this instruction is NOT valid within a transfer file**, as it will cause the file itself to stop running.

The purpose of the SFile instruction is to allow the operator to enter several lines of instructions to the transfer file without having to place an up arrow (^) as the first character typed on each line.

<u>EVERY LINE TYPED</u> after the SFile instruction will be placed in the transfer file until the EFile (Section 4-42) instruction is found.

EXAMPLE: *We wish to enter 5 lines to the transfer file in a block.*

SFile	*Signals that the following lines are to be placed in the transfer file.*
***Comment1**	*These*
:J,36	*5*
***Comment2**	*Lines*
:+3,1	*Will*
:V3	*Go into the transfer file*
EFile	*Places the System in a normal entry mode*

NOTE: Once the SFile instruction is entered, <u>ALL</u> data typed to the System will go to the transfer file until EFile is entered. <u>DO NOT</u> use SFile until you are ready to start sending instructions to the file. <u>DO NOT</u> type anything you do not want stored in the file after you type SFile.

TURN OFF MULTIPLE TRANSFER FILE LINE ENTRY

Section 4-42

Format: **EFile**

The EFile cancels the SFile condition (See Section 4-41). EFile signals the System to return to its normal mode and interpret data typed in normally.

EFile is valid within a transfer file, but is a useless instruction since once SFile is read, the System will not run a transfer file until EFile is typed in.

SUBROUTINE FILES

Section 4-43

Format: **SUb,x**

x = The line number in the file to start running

The SUb instruction allows you to define the line number where a file will start running when a GEt instruction is executed from a running transfer file. SUb has no effect if entered in the terminal keyboard entry mode since TFile must be used to start the file.

Normally the GEt instruction, when encountered in a running transfer file, will load the file and start it running at line #1. SUb may be used to cause execution at a line other than line #1.

Once a GEt is executed, SUb is reset to 1.

EXAMPLE: *We are running FILE1 which we want to load FILE2 and run FILE2 at line #15*

File #1 Contents	
.	*Various instructions*
.	
SUb,15	*Setup next file we get to run at line 15*
.	*Other instructions are OK here if desired*
GEt,FILE2	*Loads FILE2 which runs at line 15*
.	*Line 15 in File2 executes and Sub is reset to 1*
.	*Line 16 executes*
.	*etc.*
.GEt,FILE3	*Load File3*
.	*File3 executes at line 1 since no new SUB was issued*
.	

PUT LINE CURRENT TRANSFER FILE LINE NUMBER IN GLOBAL

Section 4-44

Format: **:Lx**

x = A Global number 1 to 9

This instruction will load the current file line number into the Global entered. This is valuable when used in conjunction with the SUb instruction to allow for a return to a file from another file and with the JUmp Indexed (:I) instruction.

SUb EXAMPLE: *We want FILE1 to load FILE2, run it and when it finished FILE2 IS TO RELOAD FILE 1 and allow it to continue where it left off.*

File 1 Contents

	.	*Various instructions*
LIne #20	**:L6**	*Put the CUrrent line # in Global #6 (value=20)*
	.	
Line #21	**GEt,FILE2**	*Load File2 and run it*
	.	*File loads*

File 2 Contents

	.	*Various instructions*
	:+6,2	*Add 2 to Global #6*
	:X,SU/,6	*Enter SUb instruction with value of Global 6 (value now 22)*
	GEt,FILE1	*Load File 1*

File 1 Contents

Line 20	**:L6**	
Line 21	**GEt,FILE2**	
Line 22	***FILE1 RUNNING**	*Starts executing this line when loaded because SUb/22 was received from File2.*

INDEXED JUMP PER GLOBAL VALUE

Section 4-45

Format: **:Ix**

x = A Global number 1 to 9

The indexed jump is used to alter the sequence of execution of a file based on the value of a Global.

EXAMPLE: *We have 4 named files in memory we want to load and execute the file indicated by the operator.*

FILE CONTENTS

Line #

1	***CIRCLES-Enter 1**	*Display*
2	***SQUARE-Enter 2**	*Info*
3	***DIAMOND-Enter 3**	*To*
4	***CURVE-Enter 4**	*Operator*
5	**:V1**	*Read operator response into Global #1*
6	**:Z1**	*If input 0 then skip to line 8*
7	**:J**	*If not 0 skip the next line 8*
8	**//**	*Transfer out if 0 entered*
9	**:S2,5**	*Set Global 2 to 5*
10	**:<1,2**	*Skip if # entered less the Global 2 (5)*
11	**:J,1**	*If greater than 4 entered then go to Line 1*

-here the number entered as Global 1 is greater than 0 and less than 5

12	**:I1**	*Jump to the current line plus the value of Global #1*
13	**GEt,CIRCLE**	*Jump here if Global 1 = 1 (Line 12+1)*

14	**GEt,SQUARE**	*Jump here if Global 1 = 2 (Line 12+2)*
15	**GEt,DIAMOND**	*Jump here if Global 1 = 3 (Line 12+3)*
16	**GEt,CURVE**	*Jump here if Global 2 = 4 (Line 12+4)*

Thus the proper file will be loaded and executed.

TEST TWO GLOBAL VALUES FOR EQUIVALENCE

Section 4-46

Format: **:E,x,y**

x= Global number 1 to 9

y = Global number 1 to 9

This instruction will cause the file to skip the next line if the values associated with the two Globals are equal, otherwise, (unequal) the next line is executed.

EXAMPLE: We wish to run a file until Global #2 and #6 are equal to each other.

Line #

1	**:V6**	*The operator enters a value for Global 6*
2	**:S2,1**	*We set Global 2 to 1*
.	**.**	*Various Instructions*
20	**:+2,1**	*Add 1 to Global 2*
21	**:E2,6**	*Skip if Global 2 equals the value entered for Global 6*
22	**:J,3**	*If not equal to go Line 3*
32	***DONE**	*If equal Print *Done*
33	//	*And exit*

Note: Good programming practice would insist that the Global entered by the operator be checked for range (greater than 0 and less than some limit) before starting to run in order to assure proper operation. In the example above, if the operator entered 0 then the file would run 32767 times!

PRINT THE VALUE OF A GLOBAL

Section 4-47

Format: **:P,x**

x= Global number 1 to 9

This instruction causes the value of a Global to be displayed on the terminal's display.

EXAMPLE:

Line #

1	**:S1,1**	*Set Global 1 to 1*
2	**:P,1**	*Print its value on the Terminal*
3	**:+1,1**	*Add 1 to its value*
4	**:J,2**	*Jump to line 2 and print new value*

This will print the value of Global 1 until REVISE is pushed to stop the File.

COPY THE VALUE OF ONE GLOBAL INTO ANOTHER

Section 4-48

Format: **:@x,y**

x = Global number from 1 to 9 to receive the value of Global y

y = Global number 1 to 9

This instruction allows you to copy the value associated with one Global into another. The source value (the one to be copied) is not affected.

EXAMPLE: *We wish to place the value of Global 9 in Globals 5 & 3*.

.

.

:@5,9 *Put the value of Global 9 in Global 5*

:@3,9 *Put the value of Global 9 in Global 3*

At this point Globals 3,5, and 9 all have the same value the <u>previous values of Globals 3 and 5 are lost.</u>

MULTIPLY TWO GLOBALS

Section 4-49

Format: **:Mx,y**

Where x and y are the Globals from 1 to 9 which are to be multiplied together.

This instruction multiplies the value associated with two Globals and **places the results in Global 9.** The previous value of Global 9 is lost. The values of the two multiplied Globals is unaffected.

EXAMPLE: If the following is true:

Global #2 = 5

Global #3 = 10

Global #9 = 0

Then the instruction

:M2,3

Will cause Global #9 to equal 50

Global #2 still equals 5 and Global #3 still equals 10

:M2,2

Causes Global 9 to change to 25

:M3,3

Causes Global 9 to change to 100

:M9,9

Causes Global 9 to remain at 0 (0x0=0)

ADD THE VALUES OF TWO GLOBALS

Section 4-50

Format: **:Ax,y**

x and y = Two Globals from 1 to 9 whose values are to be added together.

This instruction adds the values of the two indicated Globals **and places the result in Global 9.** The values of the added Globals are unchanged. The value of Global 9 is replaced by the sum.

SUBTRACT TWO GLOBAL VALUES

SECTION 4-51

Format: **:Nx,y**

x = Global 1 to 9 from which the other Global's <u>value</u> is subtracted

y = Global from 1 to 9 whose <u>value</u> is to be subtracted from x.

This instruction subtracts one Global value from another and **<u>places the result in Global 9.</u>** The previous value of Global 9 is lost. The values of the two Globals operated on are unchanged.

EXAMPLE: If the following conditions are true-

Global 9 = 10

Global 2 = 5

Global 4 = 2

The instruction

:N2,4

Causes Global 9 = 3 (5-2=3)

Global 2 = 5

Global 4 = 2

DIVIDE ONE GLOBAL BY ANOTHER

Section 4-52

Format: **:Dx,y**

x = Global number from 1 to 9 whose value is the divisor

y = Global number from 1 to 9 whose value will be divided.

This instruction divides the value associated with one Global by the value associated with another and puts the **truncated integer result in Global 9.**

The division and result of the division is in integer format and the result is truncated **NOT ROUNDED** to the nearest number. If 10 is divided by 3 the result would be 3.33 but the truncated integer result will be 3. **The fractional part of the result is discarded**.

EXAMPLE:

:S1,3	Set Global 1 to 3
:S2,10	*Set Global 2 to 10*
:D1,2	*Divide Global 2 by Global 1 (10/3)*

Global 9 now has the value 3 (the .33 fraction is discarded)

:D9,1	*Now divide Global 1 by Global 9*

Global 9 now has the value 1

BLIND TRANSFER

Section 4-53

Format: **BT**

When the BT command is read from the keyboard or the Transfer File, commands run from the transfer file are not printed on the terminal screen. This "Blind" mode causes the file to execute faster. It is recommended that the Blind Transfer mode be used only after the file has been proven to be error free.

NORMAL TRANSFER

(See: BT, Blind Transfer)

Section 4-54

Format: **NT**

Removes the "Blind Transfer" mode and causes the transfer file to operate normally, printing commands on the terminal screen as they are executed by the System. This instruction is not required unless BT (Blind Transfer) has been previously requested.

BACKUP A COPY OF GLOBAL VALUES

(See: Restore Globals)

Section 4-55

Format: **:B**

The Backup Globals command ':B' copies the value of all 9 Globals to backup copies. This command allows you to Backup (copy) the Global values, then change the Global Values and with another command (:R) restore all 9 to the state set when the :B command was issued. The saved copies are not affected by the ZGlobal (Zero Globals) command.

RESTORE GLOBALS FROM BACKUP COPIES

(See: Backup Globals (:B))

Section 4-56

Format: **:R**

This command restores <u>all 9</u> Global values to the values they had when the Backup (:B) command was issued. When the :R command is executed the contents of the Backup copies is unchanged.

EXAMPLE: *Set a value in a Global - save that value - change the Global then restore the original value.*

:S1,10	*Establishes 10 as the value of Global 1*
:B	*Copy all Globals the Backup*
:S1,0	Now *set Global #1 to 0*
:P,1	*Print value of Global 1*
Global #1 = 0	*Value Printed*
:R	*Restore all Globals*
:P,1	*Print value of Global 1*
Global #1 = 10	*Value Printed*
:+1,1	*Add 1 to Global 1*
:P,1	*Print Value*
Global #1 = 11	
:R	*Reset all Globals from backup*
:P,1	*Print value of Global #1*
Global #1 = 10	*Restores value last copied to backup*

GET THE BACKUP COPY OF A SINGLE GLOBAL

(See :B, :R, :F)

Section 4-58

Format: **:Gx**

x = Global # from 1 to 9

This command retrieves the value of the Global at the time the last backup copy was made. This is similar to the Restore Globals from Backup Command (:R) except that **only a single Global** copy is retrieved, the other 8 Globals are not accessed. The value returned will be the same whether it was saved with the other 8 Globals using the :B command or alone using the :F command.

BACKUP A COPY OF A SINGLE GLOBAL
(See :B, :R, :G)

Section 4-57
Format: **:Fx**

x = Global Number 1-9

This command copies a single Global to a backup location, the original Global value is not changed. This command is similar to the :B (Backup all Globals) except that only a signle Global value is changed. The Save Backups are used for a given Global value whether or not it is copied by itself or with the other 8 Globals (using the :B command).

EXAMPLE:

:S2,1	*Set Global #2 to 1*
:F2	*Save a copy of it*
:+2,1	*Add 1 to it*
:P,2	*Print current value*
Global #2 = 2	
:R	*Restore all Globals from backup*
:P2	*Print current value*
Global #2 = 1	*Value reset to that at the time of the last backup*

PUT SYSTEM VALUES INTO GLOBALS

Section 4-59

Format: **:Ky,x**

y = System value number 1-9

x = Global #1-9

This command allows you to access certain system data directly to Globals. The values of the positions of the first 7 motors, current, and target frame may all be placed into Globals.

The System value numbers are as follows:

1	Zoom Counter Reading
2	North/South Counter Reading
3	East/West Counter Reading
4	Rotation Counter Reading
5	Peg 1 Counter Reading
6	Peg 2 Counter Reading
7	Peg 3 Counter Reading
8	System Current Frame
9	System Target Frame

When the :K command is received, the value request is placed in the Global indicated.

EXAMPLE: *Put the System target frame in Global 1 and the current zoom position in Global 5*

:K9,1 *Target frame is System #9*

:K1,5 *Zoom position is System #1*

At this point the System target frame number is copied in Global #1 and the zoom counter reading is in Global #5.

LOG ALL OPERATOR INPUT TO FILE

(See OFF)

Section 4-60

Format: **ON**

The ON Command is a system command and will cause no action if contained within a file.

When the System receives the ON command via the terminal keyboard, all subsequent data entered on the keyboard will be copied to the transfer file. The only information not copied is:

1) lines deleted by the operator (using Delete or Rubout)

2) security code entries

The copying begins at the next line to enter on the file and continues until the file is filled. When 59 lines have been copied the System will issue an error message on receipt of the 60th line.

At this time **you must issue an OFF command** and either save the file (NAme) and/or zero the file (ZFile) before issuing another ON command.

To repeat a sequence of commands from the file**, first issue an OFF command** then transfer to the appropriate line in the file.

Note that normally the file should be zeroed (ZFile) before the first ON is issued.

TURN OFF LOGGING OPERATIONS

(See ON)

Section 4-61

Format: **OFF**

OFF causes suspension of the logging operation. OFF has no effect unless ON was issued and the logging operation was in process.

SCAN, Slitscan/Streaking

Section 5-1

General Information

The Cinetron System is capable of producing an almost unlimited variety of Slitscan and Streaking effects.

Both Slitscan and Streak effects are photographed with the camera movement 180˚ out of sync. This causes the film in the camera shutter to be placed in the photographing position, and causes the main (not capping) shutter to be in a position where, if the dissolving shutter is open, light will pass through the lens to the film. The System capping shutter is used to control when light passes to the film and when it is held back. As the Zoom or compound is moved with the shutters open, the images (normally under lit) in the path of the lens are streaked onto the film. **Please note that the Cinetron Animation Systems do not automatically control the main camera in and out of sync.** The sync - 180˚ positions must be set manually before starting. The capping shutter should be capped before placing the camera out of sync (180˚) to avoid exposing the first frame of the film accidently. Once the Scan operation starts, the System will control the capping shutter for you.

Obtaining the optimum exposure in this mode is generally a process of experimentation and experience. Sometimes small changes in the effect can create large exposure changes, and visa-versa.

A good starting point for your initial Slit Scan test would be with a 250 foot-candle under light and Eastman 7247 or 5247 film at about f5.6.

-----The Interrelation of Normal and Scan Modes----

Slitscan/Streak routines all work in addition to normal animation type motion. If a normal animation move occurs on a given frame and if the system CUrrent frame is set at that frame, then the normal movement will occur. If the Scan mode is being used it will also occur.

It might be helpful to consider that the Scan creates an image based on the Scan instructions given. This image can be thought of as though it was a piece of artwork. The normal instructions move that piece of artwork around if desired. The image is created by the Scan and moved around as though it were a solid object.

The Scan effect created is determined by the contents of a DAta buffer in the system memory. The size of this buffer (number of lines) varies depending upon the version of Scan you are using (Series II,III,IV).

The buffer is organized as followss:

	Distances --------------------------							Directions ------------------------------------						
Line #1	Z	NS	EW	Ro	P1	P2	P3	Z	NS	EW	Ro	P1	P2	P3
	"	"	"	"	"	"	"	"	"	"	"	"	"	"
LIne #X	"	"	"	"	"	"	"	"	"	"	"	"	"	"

Although this is not the way this "looks" in memory it is the way the computer "sees" the data. Naturally, more information than that represented is used by the system.

When ZEro is typed the system sets a "STOP SCANNING" flag (system generated) in the first position of their data buffer. This prevents the SCan from running.

Note that ZEro does <u>not</u> clear the buffer to 0, it just places the "STOP SCANNING" identifier so that scan is prevented.

When the SCan mode starts running, it UNcaps the shutter and reads the first line in the data buffer. If the "Stop Scanning" flag is not present, it moves the axis distances and directions indicated.

The SCan keeps reading and running, line after line, until the "Stop Scanning" flag is found. The shutter is then capped, and the system returns to the point where it started running the DAta. When this origin is reached, the system automatically reverts to a "Normal" mode and checks to see if there are any conventional moves given for the CUrrent frame. If no conventional moves are found and if the SHoot target frame has not been reached, a frame of film is advanced and the system CUrrent frame is advanced (while capped), and the whole process restarts.

Several different commands define the data in the SCan buffer. Refer to the various instructions for their specific effects.

SCAN

Section 5-2

Format: **SCan**

The SCan command puts the system in a SCan (Slitscan/Streak) mode. Some SCan mode instructions "TURN ON" the mode so that it is not always necessary to enter this command.

To remove SCan mode enter: ZEro or OMX

Actions of the Terminator in SCan Mode

Important Note:

If "ENd", "ERewind", "AGain", "REverse" or "ROtate" are used as a Terminator, while the Scan mode is being used, the system will CAP the shutter, run 5 frames of leader, and turn off the SCan mode when the SHoot target frame is reached. This is done to allow for a job which is to finish unattended. This prevents the last frame exposed from sitting in the gate where it might be accidentally fogged. <u>**Take note of this behavior especially when ERewind is used.**</u>

DATA

(See ZD, SCan, PData)

Section 5-3

Format: **DAta, *Zoom dist, {N/S dist, E/W dist, Rotation dist, Peg 1 dist, Peg 2 dist, Peg 3, Zoom dir, N/S dir, E/W dir, Rotation dir, Peg 1 dir, Peg 2 dir, Peg 3 dir}***

NOTE: The entire DAta command must be entered on a single line, and at least one entry must be made.

The DAta command is used to place DAta in the Scan (Slitscan/Streak) DAta buffer. Several Routines (LAse, JAm, FIgure, MFigure) also place DAta here so it is not always necessary to use this command to fill the DAta buffer. The entry of a DAta line describes all or a part of an image to be created by the SCan mode. DAta is entered by the system into a buffer which may be as short as 3 lines (entries) or large as 247, depending upon the Series Slitscan provided. Series II a,b&c SCan allow only 3 data lines to be entered where Series III and IV allow 247. When the buffer is filled the System will respond: **SCAN BUFFER FILLED.** If another DAta command is entered, the System will reset (see ZEro) the DAta buffer and start over at the first line.

When a line of DAta is read by the System, it is placed on the next line of the buffer, then the System places a "Stop Scanning" flag on the following line and sets the "DAta Write Pointer" (see PData) to write any subsequent Data on the next position in the buffer.

Appendix "G" contains the entries to be made for the directions of the axis entered. When a line of DAta is received, the system sets the line to all "0's", so any entry not made for a particular axis is assumed to be 0, as are any directions not entered. The DAta command is "position sensitive" where the order of entries is concerned. That is, the 4th number read is always treated as ROtation distance, the 7th as a Peg3 distance, etc., **if you wish to skip a motor, enter 0 in its position.**

EXAMPLE: *We wish to enter a distance of 100, and a direction of 0 on the Zoom.*

DA,100 *Everything else assumed to be 0*

To enter a distance of 300 and E/W direction 1

DA,0,0,300,0,0,0,0,0,0,1

As stated in the General Information on SCan (Slitscan/Streaking), the lines of DAta are read and executed sequentially. The DAta read at a given instant in time determines what is being moved at that time. If the Zoom had a distance to run on Line #1 but not Line #2 and if the E/W had no distance to run on Line #1 but on Line #2, then first the Zoom would run, then it would stop and the E/W would run.

EXAMPLE: *Start the N/S and E/W at the same speed, when they have moved 100 units, move the N/S at 1/2 the E/W speed, and move the E/W 100 more units.*

DA,0,100,100 *N/S - E/W both move 100 units (direction assumed to be 0)*

DA,0,50,100 *E/W goes 100 more units while the N/S moves only 50.*

To reset the DAta buffer to line #1 and place a "Stop Scanning" flag as the first entry type: ZEro. To clear the DAta buffer to all 0 and set the "Stop Scanning" as the first entry enter: ZData.

NOTE: ZEro has other effects (see ZEro). If ZData is used RScan cannot recover DAta.

Sometimes the distance entered on a DAta line will exceed the 32767 maximum size number because the scaling of the axis (See DScale, VEeder). If this occurs, the system will respond: **Amount too large - split it.** If this happens, try dividing all the entries on the DAta line by 2 or 3 and enter on 2 or 3 separate lines.

EXAMPLE:

DA,4000,200	*DAta entered*
Amount Too Large-Split It	*System responds*
	One or more axis, when scaled exceeded the 32767 limit
DA,2000,100	*Divide all by 2*
DA,2000,100	*And enter as 2 lines*

SERIES IIC DATA ENTRIES

Section 5-4

Format: **{The same format is used for the DAta command in Series IIC SCan.}**

The general concept and operation of the DAta command are virtually the same as described for Series III or IV with **several important exceptions**.

1. There are only 3 DAta lines available.
2. ZData should be used before the first DAta entry.
3. If a motor entry on a DAta line is skipped (0), the previous entry for that motor is used. That's why ZData should be used before the first DAta entry.

PData, RScan, FIgure MFigure are not implemented in Series II a,b&c, SCan.

ZERO DATA

(See ZEro, RScan, DAta)

Section 5-5

Format: **ZData**

ZData sets the DAta buffer to all 0's, and places the "Stop Scanning" flag on the first line of DAta.

IF ZData is used RScan will not restore the DAta.

ZData is primarily a Series IIC command and serves no useful purpose in Series III or IV.

POINT DATA

Section 5-6

Format: **PData,x**

x = A DAta line number from 1 to 247

PData allows the partial editing of slitscan DAta. PData is defines the next line on which the system will enter DAta. If the line number is greater than the line on which the System is already set to write, the PData will set to write on the indicated line as requested but the system will not "see" the DAta since the "Stop Scanning" flag (See Scan General Information) will still be on the line following the one the system was using. So, when that line is reached, the SCan will stop before reading the new DAta.

The intended purpose of the PData command is to reset the DAta to write on a lower numbered line than the one the system is using. By this method the end of the SCan DAta can be "Edited" or rewritten from a specific line, avoiding going to "ZEro" or "ZData" and rewriting the entire DAta buffer to correct an error.

Primarily, PData is of use when the System is being instructed from an external computer or transfer file.

EXAMPLE: *35 lines of DAta have been entered and we wish to change lines 32, 33 and 34.*

PD,32	*The system was set to write on line 35 next - we reset it to write on line 32 next.*
DA,.....	*Re-enter line 32 - place Stop Scanning flag on 33.*
DA,....	*Re-enter line 33 - place Stop Scanning flag on 34.*
DA,.....	*Re-enter line 34 - place Stop Scanning flag on 35.*

LASE

(See TUbe, SCan, DAta, ZData)

Section 5-7

Format: **LAse, Distance, # of Sections, Motor #, Motor Direction**

The LAse command is a Series II SCan command which is of little value in Series III or IV Scan since they have more efficient methods for creating the same effect..

LAse is used to describe a TUbe-type motion and is normally used in conjunction with the TUbe command in Series II SCan.

The distance entered is divided by the number of sections entered. The System divides the distance by the number of sections and displays the true distance the LAse will run. The distances displayed will not always be the same distance entered because all divisions will not be equal. The operator can re-enter the LAse using different distances until the desired result is achieved. Each section entered will be treated as frame by the TUbe command. If, however, TUbe is not used, the entire LAse will occur on each frame (provided SCan was entered). **It is NOT recommended that DAta be used with TUbe.**

Sometimes the distance, when divided by the number of sections, will be too large for the system to use. If this occurs, the system will respond**: LAse amount too large - split it**. If this occurs, either use more sections (divide by a larger number) or use a smaller distance.

The motor number entered is the motor number identifying the axis to which you are applying the entered distance. Appendix "C" contains the list of axes and the corresponding motor numbers.

The direction is the direction that the LAse is applied on the given axis. Appendix "G" contains a list of axes and corresponding directions.

THE LAse COMMAND MAY NEVER BE USED IN CONJUNCTION WITH A DAta COMMAND

It is highly recommended that ZData be used before the first LAse command is given for a job.

LAse commands accumulate or replace data on each axis as they are entered. That is, if the motor number for a LAse command is the same as a previous LAse command the data are replaced for that axis. If the motor number is different, then the DAta are added.

Note that the number of sections for each LAse must agree on a particular job or erroneous results will occur.

EXAMPLE: *We wish to produce a figure comprising of 50 N/S units and 150 E/W units to be used with a TUbe command containing 50 units (frames) per expand or collapse.*

ZData	*Clear data in SCan*
LA,50,50,2,0	*North LAse 50 units in 50 sections*
True Distance=50	*System response*
LA,150,50,3,1	*West LAse 150 units in 50 sections*
True Distance=150	*System response*

Note that a TUbe command would also have to be entered.

EXAMPLE: *Same parameters as in the first example except that we wish to use 26 sections.*

ZD	*Clear data in SCan*
LA,50,26,2,0	*North LAse 50 units 26 sections*
True Distance=49	*49 distance is response from system*
LA,51,26,2,0	*Se we re-enter with the*
True Distance=50	*distance as 51 units*
LA,150,26,3,1	*West LAse 150 units 26 sections*
True Distance=149	*149 is the system distance response*
LA,151,26,3,1	*So we try a distance of 151*
True Distance=150	*Now the System distance is 150*

EXAMPLE: *LAse sections must be equal for a given job.*

ZD	*ZEro DAta*
LA,150,10,1,0	*OK so far, LAse up 150 units in to sections*
True Distance=150	*System responds*
LA,150,20,2,0	*In itself would be OK, but since the first entry was 10 sections* ***this will cause an error***

In this example the result would be that the entry with the greatest number of sections (20) would cause the entry previously made to be recopied into 20 sections. The Zoom (first entry) would run 300 units (20/10=2 --- 2X150=300 units) and the N/S 150.

EXAMPLE: *LAse sections must be the same for a given job.*

ZD	*Clear data*
LA,150,20,2,0	*OK*
True Distance=150	
LA,150,10,1,0	***Causes an error** because sections =10 not 20*
True Distance=150	

In this example the second LAse section entry was shorter by 50% than the first. If run the N/S (the first entry) will only run 10 sections or 1/2 the distance (75) while the second entry for the Zoom axis will run the full 150 units. This is because **the system will use the number of sections in the last lase entry to run all the data entered.** When 10 sections are run the SCanning stops.

TUBE

(See DAta, SCan, LAse)

Section 5-8

Format: **TUbe, # of frames to expand, # of frames to hold, # of frames to collapse**

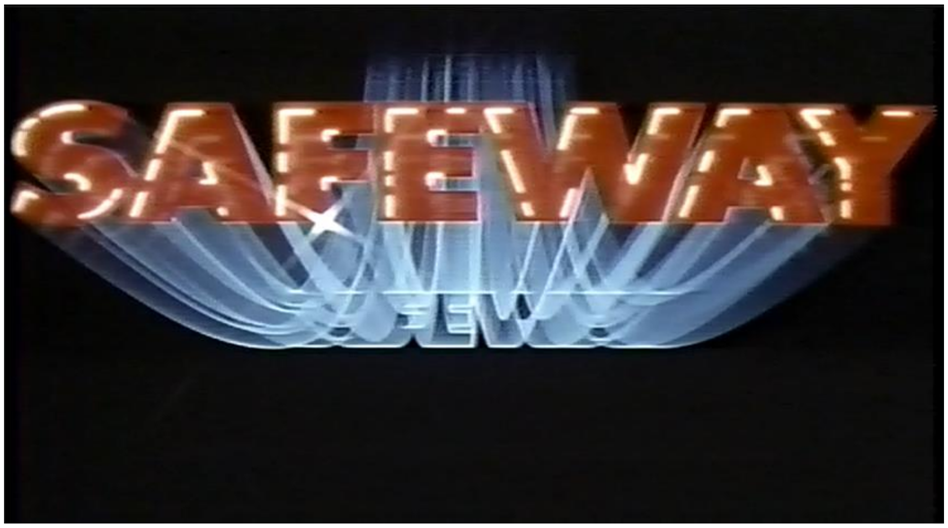
TUbe automatically sets the system into the SCan mode (See SCan).

TUbe indicates how the LAse is to be executed. **In Series III and IV it indicates how the DAta is to be accessed.** (If your system operates under Series III or IV SCan the following references to the “LAse number of sections”should be read as: "DAta lines".)

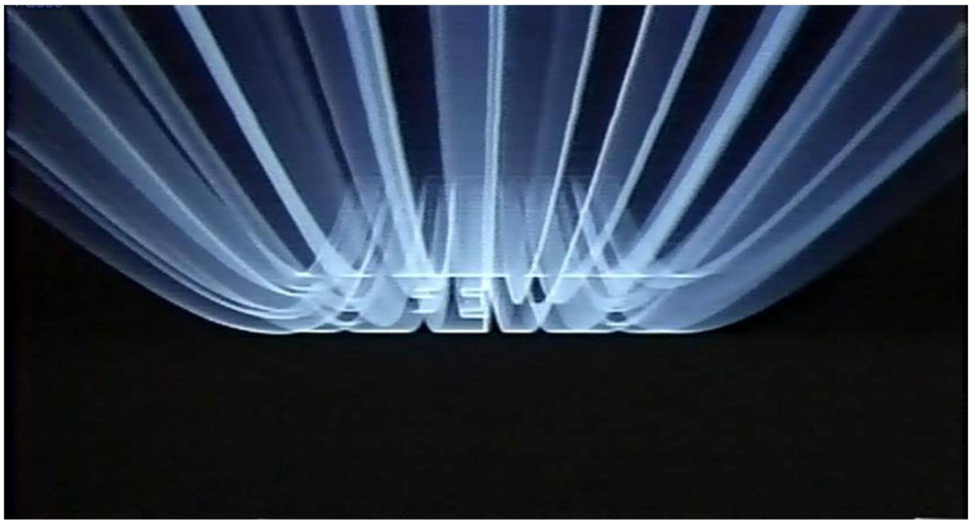
The number of frames to expand or collapse must divide equally into the number of sections entered in the LAse command or, both may be 0. IF the division is unequal, an error will be displayed by the System when the TUbe is run.

The number of holding frames indicates how many frames (**plus 1 frame**) to hold the expanded tube. The number of holding frames must be at least 1 frame. (TU,4,1,4 will shoot 8 frames.)

The TUbe will run one section of the LAse for each frame at the Expand, adding a section on each subsequent frame until the described figure is fully "Expanded". Then, the System will "Hold" that figure for the number of frames indicated, then "Collapse" the figure one section at a time from its origin until totally collapsed. **Note that this entire sequence will occur regardless of the System shoot target frame.**



(Figure TU-1)



(Figure TU-2)

In the figure TU-1 above, there is an illustration of the Safeway™ logo. The logo appears to sweep down from the top, through a roughly centered position then sweep out of the top of the screen while its path is left as a trailing blue outline as seen in figure TU-2. The outline, created as a tube, collapses as the logo moves out of the top of the frame. In the finished product, the logo is re-introduced at the center position.

We used the following technique to create the effect:

First, we create a conventional zoom lasting the length of the desired effect that will zoom down while panning the table first north then, once it passes the position where the logo is centered, south to a point where the logo is just out of the top of the frame. (Remember that the Cinetron system considers a direction to be that which the Centerline of the reticle moves. South means make the center of the frame move south.)

MFig (Mental Figure) is used in Series III or IV scan to create the scan data for the tubing action desired. The TUbe command is issued to create the first pass using an under lit outline of the logo. The film is then rewound and the solid logo front and sparkle and 3-d effects are added by making multiple under lit passes.

ZEro	*Clear system commands*
ZData	*Clear data for scan*
LOck	*Memorize start position*
CU,1	*Set current frame*
DOWN,5000,1,61,1,1	*Describe motion path*
SOUTH,760,1,31,1,6	
NORTH,210,31,61,6,1	
MFig	*Mental figure data*
GO,61	*to frame 61*
CUrr,1001	*Set frame count outside move used to MFig*
TU,60,2,60	*TUbe 60 sections (Turns on Scan)*
Set capping shutter closed and shuttle movement to 180°	
GO	*Shoot tube*
Set shuttle movement to normal	

OMX	*Turn off scan*
INdex	*Return to head position of move*
GR,1001	*Rewind film to head of effect*
CUrr,1	*reset current frame*
0Remainder	*Reset remainders to be extra tidy*
Set shuttle to normal …	*Setup art for first Front run*
Shoot,61	*Target for front art zoom*
GRewind	*shoot & rewind to first frame*
INdex	*Return to head of move*
AGAIN	*Repeat the index and again and rewind to frame 1 for each run to build up the front of the logo*

TUBE ROUTINE MODIFICATION

(See TUbe, DAta, SCan)

Section 5-9

Format: **2Tube,x,y,z,zz**

x = CUrrent frame number to Stop TUbe Expand

y = CUrrent frame number to REstart TUbe Expand

z = CUrrent frame number to Stop TUbe Collapse

zz = CUrrent frame number to Restart TUbe Collapse

2Tube is used to suspend and restart the TUbe command. 2Tube is used where a TUbe operation suspends its expansion or collapse or both. That is, with a 2Tube suspension of the TUbe, Expand the tube will expand to the given CUrrent frame and "HOLD" to the CUrrent frame given to restart. On that frame, the TUbe will continue to expand as though 2Tube had not been used.

2Tube may be entered at any time before the Terminator.

EXAMPLE: *We wish to allow a TUbe to run normally to CUrrent frame 47. Then we wish to restart the Expand at frame 59. We do not want to modify collapse*.

.	*Other instructions*
.	*Other instructions*
.	*Other instructions*
2Tube,47,59	*Suspend TUbe at 47 and hold the generated figure to frame 59, then continue the TUbe operation.*
TU,10,24,16	*TUbe description*
CU,40	*CUrrent frame set*

GO,41 *The TUbe will expand from frames 40 to 47. Then expansion will suspend to frame 59 where expansion will continue to frame 61. A hold of the full figure will occur from 61 to 84, then a collapse from 84 to 94. When done, the CUrrent frame = 94.*

RESTORING ZEROED SCAN DATA

(See ZEro, ZData)

Section 5-10

Format: **RScan**

The RScan command will restore the DAta in the scan buffer after a ZEro command has been entered. <u>If ZEro has been written more than once before RScan is used or if a DAta, LAse, JAm, FIgure, or MFigure has been used since the last ZEro, then it will not be possible to REstore the SCan DAta.</u>

The ZEro operation copies the First Value in the DAta buffer into a back-up storage area and replaces it with a "Stop Scanning" identifier. RScan replaces the value which was in a back-up storage to the first value (Stop Scan) of the DAta Buffer, effectively Restoring the SCan DAta.

EXAMPLE: *We wish to ZEro all data in the system but not the SCan DAta*

ZEro	*Tell the system to Clear*
ZEROED!	*System response*
RS	*Restore the SCan DAta Buffer*
.	
.	
.	*Continue instructions*

SCAN SPEED

Section 5-11

Format: **SPeed,x**

x = An integer value from 1 to 32767

The SPeed command allows the operator to vary the running speed of SCan/Streak routines.

Because of varying mechanical and exposure conditions, it may, from time to time be necessary to set the SPeed of a SCan operation at a higher or lower value.

Normally the speed of SCan should be around 14 with 200 step boards or around 3 with 400.

A lower number will run the SCan faster, a higher number slower.

Although the faster SCan SPeeds cause a more critical exposure situation that the slower ones, the primary intent of the SPeed command is to adjust for an axis which may be having trouble running because of mechanical difficulties.

PUSH BUTTON SCAN DATA ENTRY

(See DAta, SCan, REcord, FOllow, ZData, ZEro, PData)

Section 5-12

Format: **JAm**

The JAm routine loads the system SCan DAta buffer from the hand control switches.

The JAm routine responds:

SCAN JAM ROUTINE ESCAPE WITH REVISE

When JAm is running the system allows you to run the hand control switches. When "Continue" is pressed it writes the <u>difference</u> between the system current position and where it was when the routine started, or when the last position was "JAmmed in", and writes the DAta line number and positions on the terminal.

You may wish to think of the JAm routine as being to SCan what the REcord and FOllow routines are to normal shooting.

JAm allows you to visually follow a drawing and mark to the system, via the "Continue" button where you wish each DAta line to be entered, having the system automatically enter the distance and direction DAta.

JAm writes DAta to the next available DAta line. IT is necessary to ZEro or ZData or PData before using JAm if it is desired to start at the first line of DAta or at a line other than the next line the system would ordinarily use.

It is necessary to press "REVISE" to escape from the JAm routine.

Typical sequence of operation:

	first move to first position
LOCK	*Remember where to start*
ZEro	*Clear system and Reset Data to first line*
ZEROED!	*Response*

JAm	*Turn on JAm*
JAM ROUTINE ESCAPE WITH REVISE	*Response*

Move to second position, press "Continue"

JAm will send DAta Line # and counter readings

Move to third position, press "Continue"

JAm will send DAta Line # and counter readings

This continues until the last position has been entered then press "REVISE".

PROGRAM suspended for Revision - *System response*

At this point DAta for a SCan has been entered via the manual switches.

CU,1	*Set CUrrent frame*
INdex	*Set to return to 1st position (LOCK)*
SCan	*Turn on SCan*
Go,10	*Shoot to frame 10*

At this point the system will INdex, turn on the SCan mode and follow through the DAta points JAmmed in via the switches. JAm may be used in conjunction with DAta, FIgure or MFigure.

FIGURE SCAN

(See DAta, SCan, AUdition)

Section 5-13

Format: **FIgure**

FIgure is used to convert a conventionally described movement to a slitscan movement. For the purpose of FIgure, DAta lines may be thought of as frames in a conventional mode (or automatic move). **When the system is in the FIgure mode, it is also automatically in an AUdition mode.** The positions of each frame of a conventional move, which is AUditioned automatically, are converted and written into successive DAta lines.

EXAMPLE: *We wish to SCan through a path described as a N/S E/W series of moves.*

ZEro	*Clear all commands & Reset Data*
ZEROED!	*Response*
East,100,1,24,5,BO	*The "Normal" move*
West,200,24,48,5,BO	*Travels North from frames*
NO,300,10,48,5,BO	*1 to 48 while the E/W zigzags*
CU,1	*System CUrrent frame for figure*
GO,48	*& Audition through move and load the SCan DAta*

When the move finishes, the move description will be in the DAta buffer lines 1 to 48.

Note please that it is a coincidence that the DAta line <u>numbers</u> are the same as the CUrrent frames. *If the CUrrent had been 1000, the DAta would have still been entered at line 1 because ZEro sets the DAta line to 1.*

-- ***<u>The System Automatically Returns To The EXpose Mode when FIgure finishes.</u>***

SCan	*Turn on SCan mode*

CU,100 *Here we will set the CUrrent frame to 100 so that the original move from 1 to 48 will not be run along with the SCan.*

GO,105

The system will now scan through the path FIgured on each of the 5 frames given.

-- To TUbe the foregoing example:

Enter: TU,12,x,48 with "x" being the holding frames if desired.

The system will TUbe each of the positions, first expanding from 1 to 12, then holding, then collapsing for 48 frames.

Because of the way that FIgure operates, it is very easy to make a "Front" or "Back" burn-in pass for a streak, then using that description, FIgure the SCan and the Expand and Collapse a leading or trailing streak.

Since there are 247 DAta lines, the length of a move to be FIgured cannot exceed 247 frames.

MENTAL FIGURE

(See FIgure, RUn)

Section 5-14

Format: MFigure

The Mental Figure is exactly like FIgure except that the move is not run through in the AUdition mode. Instead, the system "mentally" FIgures all the DAta positions without moving.

It may take the system 2 or 3 seconds to Mentally FIgure a move so that **it is recommended that RUn be used as a Terminator so that the "CUrrent frame" message will be printed to let you know that MFigure is done.**

ILLUMINATE SCAN-STREAK

Section 5-15

Format: **IL,x**

x = The number of DAta lines or LAse sections to expand to.

Illuminate is used when it is desired to expand a TUbe less than its full extent and then to move that illuminated section through the path described in the DAta, LAse or FIgured entry. Illuminate works as follows:

> The System will expand the number of sections (frames) entered, then the origin of the figure will be moved towards the end of the figure 1 section and the same number of sections will be run. This continues until the end of the figure is located. Then the System will begin to collapse (Hold Frames and Collapse Frames entries are ignored) 1 section per frame until the figure is collapsed.

Using the illustrations TU-1 and TU-2 in Section 5-8 the command:

IL,20

will cause the figure to expand from the origin to the boundary between sections 19 and 20. On the next frame the origin would move up to the boundary between sections 1 and 2 and the figure would run to the boundary between 20 and 21 and then the origin would move to the boundary between 2 and 3 and the end would be between 21 and 22. This process would continue until the end of the section being illuminated reached DAta line 60.

At this point, the section illuminated would run from data lines 40 to 60 and the collapse would start. The next frame would originate at the boundary between 41 and 42 and run to the end at position 60. The collapse will continue until the end is found as the origin. In this example 20 frames would be exposed expanding, 20 frames would be used "Hunting" the end and 20 frames collapsing

WRITE MODE

Section WRITE

Format: **WRite**

The WRite command can be entered any time before a terminator. WRite causes the system to write out the positions of each axis for each frame in a conventional move. No movement occurs and no film is exposed. The WRite command can be used to create an exposure sheet that can be used on an animation stand or rail camera not equipped with a computer system or to assist in layout of drawn animated moves.

The display format of a WRite is as follows:

Frame #,Axis 1,2,3,4,5,6,7,8

WRite is removed by ZEro,EXpose and OMX

VERIFY SWITCH

CRITICAL INFORMATION

One of the manual switches that is active at all times is one labeled “VERIFY”. When this switch is on, the system will behave as follows:

IF IN THE DATA INPUT STATE whether running a Transfer File or not, the system will read data in and check it for syntax only. If the syntax is correct, the system will send an asterisk (*) to the user’s terminal and read the next line entered. No motors will move and no film will be exposed. If there is an error, the error will be reported to the terminal and the user will be required to press “REVISE’ or “CONTINUE” to proceed. The asterisks are sent just to remind you that Verify is on.

IF SWITCHED ON WHILE THE SYSTEM IS EXECUTING A MOVE the results are unpredictable.

The purpose of VERIFY is to permit the user to send a batch of commands, usually from a terminal disk file or a system Transfer File and prepared off-line. The VERIFY mode will permit quick validation of the file contents without running the motors and perhaps wasting time and film.

VERIFY does not try to “make sense” of the commands; it is possible that a command be valid and still create a bad situation. For example, forgetting to uncap or cap the shutter at the proper time or rewinding or advancing to the wrong frame or just plain nonsense commands such as running a peg track off the end or rotating the table with a track extended enough to hit a column.

APPENDIX A

FRAME COUNTING AND DATA MANIPULATION

Current	Set the system Current Frame
Shoot	Set the system Target Frame
Zero	Clear all data entered and copy to Backup Buffer
Restore	Restore all data to point where last Zero was entered
Omit	Zero (Clear) the selected Motor Buffer
Flip	Reverse the directions in the selected Motor Buffer
DScale	Scale the selected motor axis
VEeder	Scale the selected motor axis (interactive)
WMotor	Set the Motor Buffer line on which to write the next command for that axis
DBuff	Display the contents (commands) of the axis Motor Buffer

APPENDIX B

AXIS MOTION IDENTIFIERS

(Only the first 2 letters are required)

UP	Zoom Up
DOwn	Zoom Down
NOrth	Move Centerline North
SOuth	Move Centerline South
EAst	Move Centerline East
WEst	Move Centerline West
CLockwise	Rotate Table Clockwise
CCLockwise	Rotate Table Counter-Clockwise
1East	Move Peg Track 1 East
1West	Move Peg Track 1 West
2East	Move Peg Track 2 East
2West	Move Peg Track 2 West
3East	Move Peg Track 3 East
3West	Move Peg Track 3 West
4East	Move Peg Track 4 East
4West	Move Peg Track 4 West
1+	Turn Auxiliary motor #1 Shaft Clockwise
2+	Turn Auxiliary motor #2 Shaft Clockwise
3+	Turn Auxiliary motor #3 Shaft Clockwise
4+	Turn Auxiliary motor #4 Shaft Clockwise
5+	Turn Auxiliary motor #5 Shaft Clockwise
6+	Turn Auxiliary motor #6 Shaft Clockwise
1-	Turn Auxiliary motor #1 Shaft Counter-Clockwise
2-	Turn Auxiliary motor #2 Shaft Counter-Clockwise
3-	Turn Auxiliary motor #3 Shaft Counter-Clockwise
4-	Turn Auxiliary motor #4 Shaft Counter-Clockwise
5-	Turn Auxiliary motor #5 Shaft Counter-Clockwise
6-	Turn Auxiliary motor #6 Shaft Counter-Clockwise

APPENDIX C

SYSTEM MOTOR NUMBERS

Axis	Motor Number	
Zoom	1	
North/South	2	
East/West	3	
Rotation	4	
Peg Track 1	5	*Also 1100-A Color Wheel
Peg Track 2	6	
Peg Track 3	7	
Peg Track 4	8	
Auxiliary	9	
Auxiliary 2	10	
Auxiliary 2	11	
Auxiliary 3	12	
Auxiliary 4	13	
Auxiliary 5	14	
Auxiliary 6	15	*Also 1100-B Color Wheel
Fade/Dissolve	16	*only DB & WM User Commands allow access

APPENDIX D

HOOKUP IDENTIFIERS

IDENTIFIER	AXIS
HZ	Zoom
HN	North/South
HW	East/West
HR	Rotation
H1	Peg 1
H2	Peg 2
H3	Peg 3
H4	Peg 4
H5	Auxiliary 1
H6	Auxiliary 2
H7	Auxiliary 3
H8	Auxiliary 4
H9	Auxiliary 5
H0	Auxiliary 6

APPENDIX E

System INDEX / LOCK Command Identifiers

"Lock" Command which causes system current position to be memorized	**"Index" Command which causes system to return to "Locked" position**
LOck	INdex
AUto	INdex
MEmo	SEtup
1Lock	1Index
2Lock	2Index
3Lock	3Index
4Lock	4Index
5Lock	5Index
* USER INDEX	*6Index (See note below)

Housekeeping Indexer

NOTE: *6Index returns to 6Lock or "Housekeeper Index" position which is a point that is determined by the system and is not under the user's control.

APPENDIX F

Identifiers for Origin and End Index Commands used in AMove Command

Format: **OI (or EI), Identifier**

To Identify	Enter as Identifier
INdex/LOck	L
SEtup/Memo	M
1Index/1Lock	1
2Index/2Lock	2
3Index/3Lock	3
4Index/4Lock	4
5Index/5Lock	5
User Index/User Lock	U
All Safes may be used as origin or end	SA and the safe number or S and the safe number

APPENDIX G

Direction Entries for SCAN Data or Lase Commands

TO MOVE	Enter Direction As
Up	0
Down	1
East	0
West	1
Clockwise	0
Counter Clockwise	1
Peg 1 East	0
Peg 1 West	1
Peg 2 East	0
Peg 2 West	1
Peg 3 East	0
Peg 3 West	1

APPENDIX H

Manual Control Panel Switches

"ENABLE/DISABLE" is always active and will prevent or enable another frame from being exposed by the system dependent upon whether "ENABLE"d or "DISABLE"d from shooting.

The following are active **only** when the system is in a "WAIT" state

---

Manual Indexer Switches:	Lock and Index
	Memo and Setup
	1Lock and 1Index
	2 Lock and 2Index
	2Lock and 3Index
	Record (Records in next available "Safe" position)
OPEN/BLACK	
ACCURIZE	
CONTINUE	(also active is error or system message)
REVISE	(Also if shooting or running Transfer File)
EMERGENCY STOP	(Active if axis motors are running)
MANUAL FADE/DISSOLVE:	(Also Active at other times see Manual Fade/Dissolve section)
IN/OUT Switch	
FADE/DISSOLVE Switch	
LENGTH (frames)	
STEP PROGRAM	

Manual Hand Controller:
Up
Down
North
South
CW
CCW
1East
1West
2East
2West
3East
3West

Inching Speed Button

The following switches are active when the system in in a RUN state

REVISE	(Also active in WAIT state)
EMERGENCY STOP	
AUDITION	
SIMULATE	
VERIFY	
CONTINUE	(active if error or encountered)

APPENDIX I

Omit and Flip Axis Identifiers

Identifier	Axis
Z	Zoom
N	North/South
E	East/West
R	Rotation
1	Peg 1
2	Peg 2
3	Peg 3
4	Peg 4
5	Auxiliary 1
6	Auxiliary 2
7	Auxiliary 3
8	Auxiliary 4
9	Auxiliary 5
0	Auxiliary 6
F	Fade/Dissolve
D	Fade/Dissolve
A	ALL AXES
X	*All Special Routines OFF

***NOTE: ALL Special routines are turned off with the OM,X command:**
Multi, Stationary Multi, Figure, Mental Figure, Pause, Perspective and Tube.

APPENDIX J

Optical Printer Motor Numbers

Motor Number	Axis
1	Camera Zoom
2	Camera North/South
3	Camera East/West
4	Aerial Zoom
5	Aerial North/South
6	Aerial East/West
7	Primary Auxiliary & Cinex Wheel
8	Auxiliary 1
9	Auxiliary 2
10	Auxiliary 3
11	Auxiliary 4
12	Auxiliary 5
13	Auxiliary 6
14	Auxiliary 7
15	Auxiliary 8
16	Fade/Dissolve