OPERATING MANUAL FOR THE ESL DISPLAY CONSOLE

R. H. Stotz and J. E. Ward

March 9, 1965

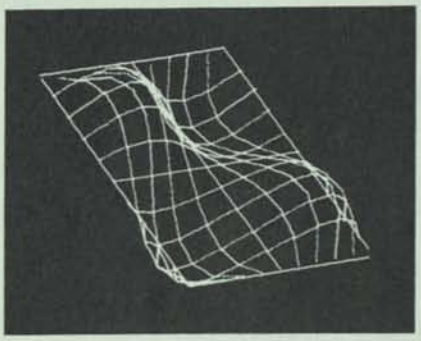
ESL Internal Memorandum 9442-M-129

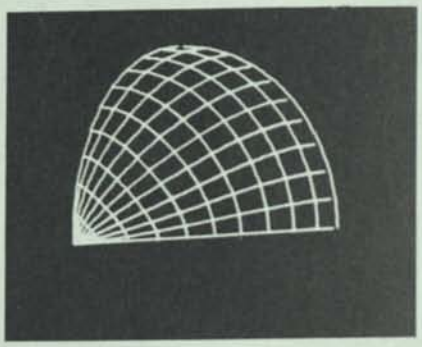
Project MAC Internal Memorandum MAC-M-217

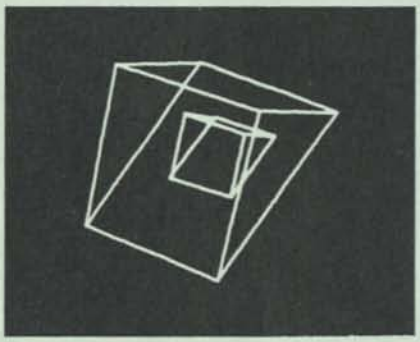
The work reported in this document has been made possible through the support and sponsorship extended to the Massachusetts Institute of Technology, Electronic Systems Laboratory, by the Manufacturing Technology Laboratory, RTD, Wright-Patterson Air Force Base under Contract No. AF-33(657)-10954, M.I.T. Project No. DSR 9442. It is published for technical information only and does not necessarily represent recommendations or conclusions of the sponsoring agency.

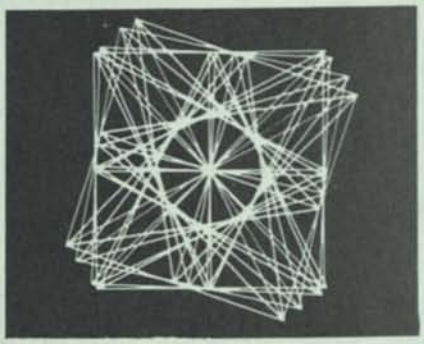
Work reported herein was supported (in part) by Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number NOnr-4102(01). Reproduction in whole or in part is permitted for any purpose of the United States Government.

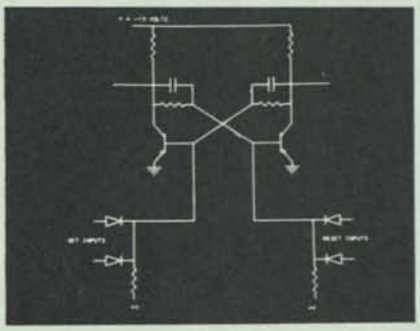
Electronic Systems Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

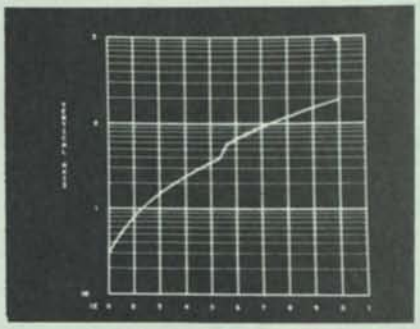












The ESL Console is a specialized computer which automatically converts three-dimensional drawing commands into arbitrary two-dimensional projections. Real-time rotation, translation, and scale change are possible even in time-sharing. Light-pen tracking is fully automatic, and either picture elements or character information may be displayed.

PREFACE

This memorandum describing the hardware functions and command structure of the ESL Console supersedes ESL Memorandum 9442-M-89 "Programming for the ESL Display Console," R.H. Stotz, August 13, 1963. It has been prepared in photo offset format because it will eventually form a section of a joint ESL-Project MAC Technical Report on graphical language techniques in time sharing. Other sections of the report, based on memoranda MAC-M-201 and MAC-M-216, will describe the A-core and B-core programming systems associated with display operation. An introductory section will describe the underlying philosophy of these complementary hardware and language developments and their relation to Computer-Aided Design.

John E. Ward
March 9, 1965

CONTENTS

CONTENTS (Continued)

LIST OF FIGURES

A. INTRODUCTION

In 1963, the Computer Applications Group of the Electronic Systems Laboratory designed and built a Display Console shown in Fig. 2.1, to be connected to the Direct Data Connection of the IBM 7094 Data Channel. Since January 1964 it has been in operation in the Project MAC Time Sharing System, and is located in Room 908, Technology Square. The purpose of the unit is to provide a direct, fast, computer-controlled display plus a flexible set of input devices including a light pen. It was designed with special attention to the needs of Computer-Aided Design under the restriction of a time-sharing system, but its flexibility makes it a useful tool with many other applications as well. The Console is an outgrowth of the Manual Intervention System which was connected to the Co-operative Computing Laboratory's 709 for several years.

The primary purpose of this document is to give an outline of the functional units in the Display Console and to describe in detail its command structure. A basic knowledge of the programming of the IBM 7094 and the Direct Data Channel is assumed. The reader is referred to the IBM 7094 Reference Manual, and IBM Special Features Bulletin for RPQ M90976 for a description of the Direct Data Channel.

In the Project MAC Time Sharing System, the Console is controlled by an Input/Output Adapter program, (the "A-Core System"), which is part of the Time Sharing Supervisor Program. Thus the user needs to be familiar with this portion of the Supervisor and its calls and procedures in order to use the Console. The current version of the Console Adapter is described in a separate document.* In addition, a "B-Core System" has been written** which consists of a set of subroutines to perform standard functions, that can be read into B-Core along with the user's program. They enable the user to think in terms of "pictures" and "picture parts", to which arbitrary names may be

---

* Bayles, R.U., "New Operating System for the ESL Display Console," Memorandum MAC-M-201 (Also Memorandum ESL-9443-M-118), December 10, 1964.

** Lang, C.A., "New B-Core System for Programming the ESL Display Console," Memorandum MAC-M-216 (Also ESL Memorandum 9442-M-216), January, 1965

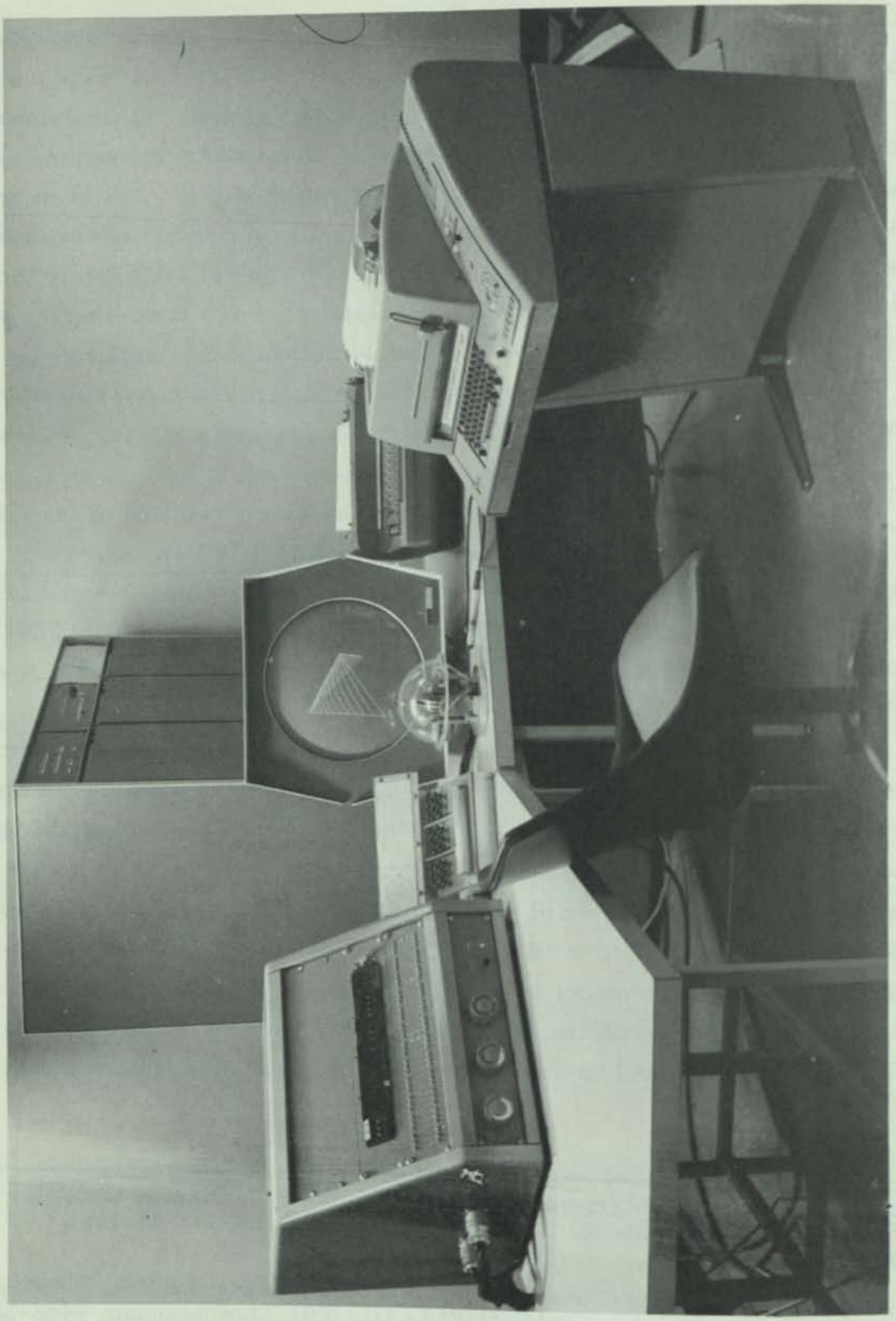


Fig. 2.1 ESL Display Console

assigned, to provide a convenient means of identifying them during communications between the A-core system and the user's programs. They also remove, in most cases, the need for working directly with both the Console commands and the A-core system.

Because this Console is being designed as a research tool, it will be prone to modification and change. As these alterations are made, programming details will change. Potential users of the Console are advised to ensure they are up to date with the latest revisions of this manual, and also of the A-core and B-core programming systems for the Console.

B. SYSTEM DESCRIPTION

1. The Console Hardware

The Display Console is shown in block diagram form in Fig. 2.2. A portion of the Console is a Type 330 Incrementing Display, made by the Digital Equipment Corporation of Maynard, Mass. to M.I.T. specifications. This consists of the magnetic-deflection cathode-ray tube with housing, table, power, and deflection amplifiers; the digital-to-analog converters; the digital registers which contain the h and v coordinates of the beam position; and a number of special controls. The scope has 1,024 unique horizontal positions and a similar number of vertical positions, thus providing over 1 million discrete points that can be specified. It has an active surface 9 3/8 inches on a side.

A unique feature first used in the Type 330 is the ability of the beam position registers (called the h and v registers for horizontal and vertical position) to count up and down at high speed. This permits lines to be drawn by introducing strings of pulses into the h and v registers and intensifying (unblanking) after each pulse. The current plotting rate in the incrementing mode is 1.8 μsec per point.

Another unusual feature ordered by M.I.T. in the Type 330 scope is three extra bits on the high-order end of the h and v registers (for a total of 13 bits). This allows h and v to be incremented off the edge of the screen without having the line appear coming on at the opposite edge (wrap around). The active (visible) scope surface is described by 2^{10} horizontal and a similar number of vertical positions. This area is called the Scope Field. The three extra bits allow the console to process lines on a field with sides 8 times as large as the Scope Field. This larger field is referred to as the Total Console

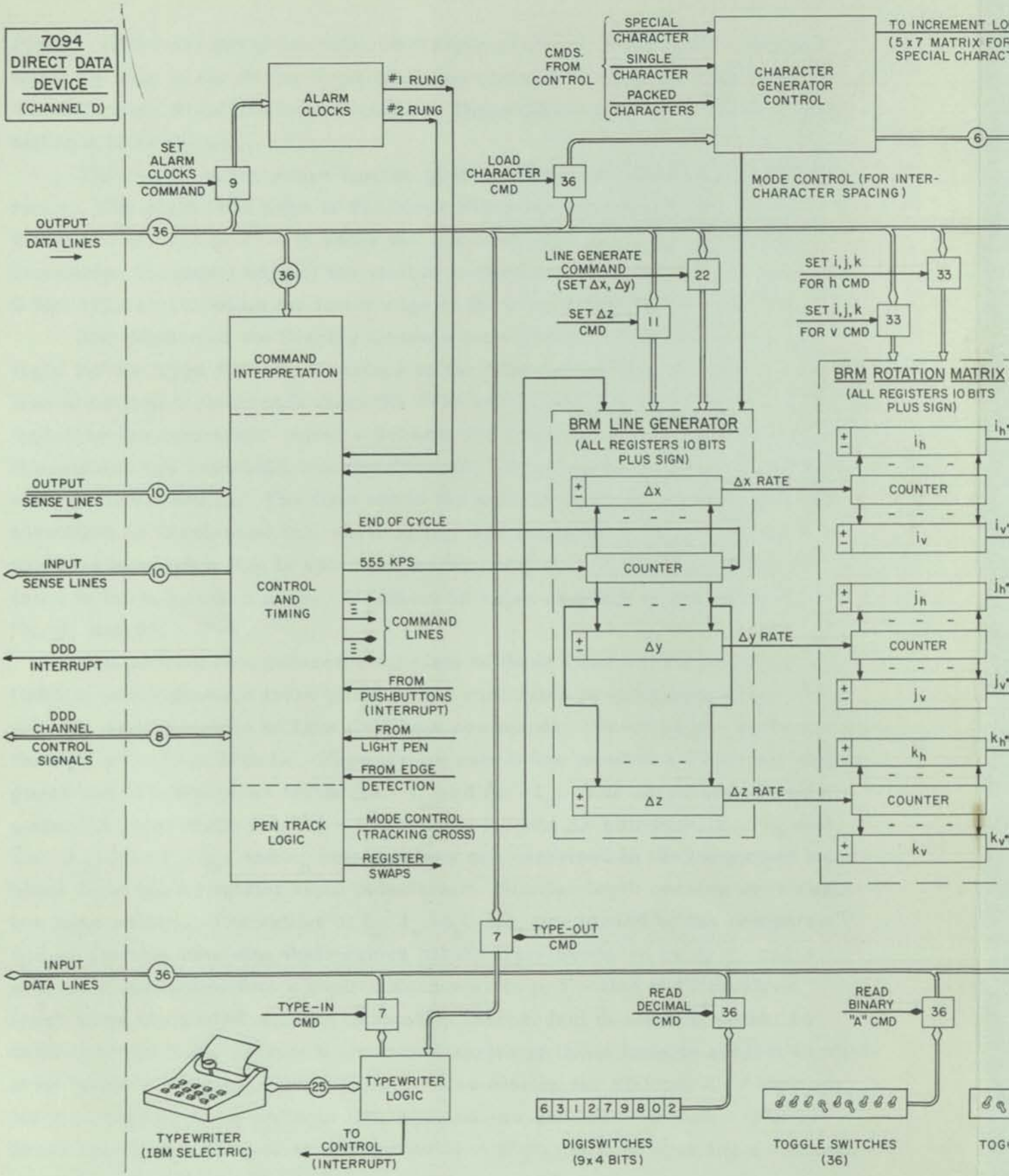
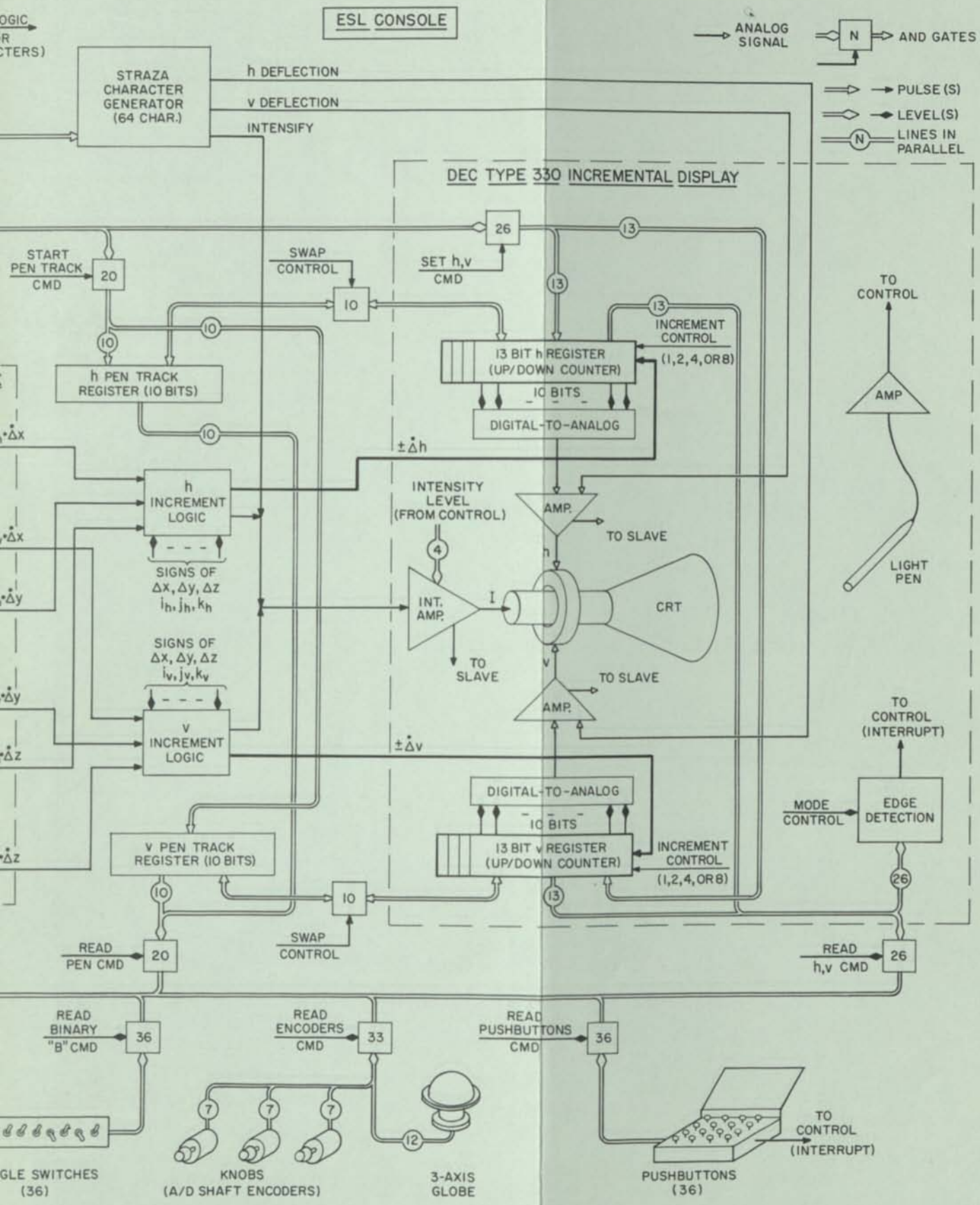


Fig. 2.2 Block Diag



gram of Console

Field. There are programmable interrupts available to alert the computer when the edge of the Scope Field has been crossed or when the edge of the large Total Console Field has been crossed. These interrupts are described in detail in a later section.

The center of the scope has the binary address 0 000 000 000 000 for h and v. The right hand edge of the Scope Field has the horizontal value 0 000 111 111 111 (= 2^9 - 1) while the left hand edge is 1 111 000 000 000. Similarly, the upper edge of the visible section has the vertical value 0 000 111 111 111 while the lower edge is the ones complement of this.

The portion of the Display Console built by ESL contains the driving logic for the Type 330, the interface to the 7094 Direct Data Channel, a section to interpret commands from the 7094 and controls to perform the function called by the command. Also, a scheme for automatic picture rotation in three dimensions has been built into the Console. Therefore, two sets of coordinate axes are referred to. The first set is the axes of the scope itself which are identified as "horizontal (h), vertical (v), and depth (d)." The second set is the axes in which a line is specified by the computer, i.e., before it is rotated to the h, v, and d axes. Distances in these axes are referred to as "x, y, and z".

The present line generator consists of three Binary Rate Multipliers (BRM), which produce three pulse trains with rates proportional to the Δx, Δy, and Δz values of Line Generate commands. These pulses are processed through a Rotation Matrix. Here the Δx rate is the input to a BRM pair which generates two new pulse trains $\dot{\Delta}x \cdot i_h$ and $\dot{\Delta}x \cdot i_v$. The Δy pulses produce a similar pulse train pair $\dot{\Delta}y \cdot j_h$ and $\dot{\Delta}y \cdot j_v$, and Δz generates $\dot{\Delta}z \cdot k_n$ and $\dot{\Delta}z \cdot k_v$. The i_h, j_h, and k_h output pulses are combined in the increment logic block to be the h register input pulse train. Similar logic creates the v register input pulses. The values of i_h, i_v, j_h, etc. are loaded by the computer, and by suitably choosing their values all the lines produced by x, y, and z may be transformed into a rotated axonometric projection of themselves. Since lines are called out incrementally, independent of starting point, an entire picture built up from a connected series of these lines is subject to whatever rotation the computer calls for. Also moving the starting h, v location for a picture made up of connected lines moves the entire picture. The rotation matrix can be used to change the size of displays by applying a common scale factor to the i_h, i_v, j_h, etc.

Pen tracking in the Console is completely automated; that is, except for initiating or halting tracking, the entire operation of generating the cross, computing the new center, and repositioning the cross, is done in the hardware. A pen-track cross is generated once every 10 milliseconds and requires about 1 millisecond to complete. When it is time to generate the cross, the command logic prevents acceptance of the next word from the Data Channel and swaps the contents (ten least significant bits) of the h and v registers with the Pen Track Registers. This saves the current display location, and puts the old pen position in the h and v registers. A block of logic associated with the line generator then causes it to produce the tracking cross, store the information as to which points the light pen has seen (producing error vector components), and add these components to the h and v registers to update the pen position. The registers then swap back again, and the command logic is allowed to proceed. The pen-track logic is unaffected by the rotation matrix. The computer may read the Pen Track Registers at any time except during a tracking cycle.

Characters are generated by either of two separate systems. A Straza Symbol Generator, which is a separate unit located under the scope table, interprets six-bit character codes to select one of 64 stored characters which match those of the KSR-35 Teletype. Diodes wired on the character cards specify a sequence of incremental deflections for up to 16 points in a 15 by 16 matrix. The Straza produces analog deflections signals for h and v which are added in the analog preamplifiers just prior to the deflection amplifiers.

The Special Character Generator, constructed by ESL, allows programmed symbols. Special logic causes it to step the scope beam through a 5 by 7 raster, and the intensification of each of the 35 points of the raster is controlled by a corresponding bit in a word from the computer. Thus, a single Special Character requires a full word to specify it. In either character generator, four character sizes are available, and space between characters (which is controlled by the line generator), is entirely programmable in size and direction. Also, both character generators by-pass the rotation matrix, i.e., characters always remain upright.

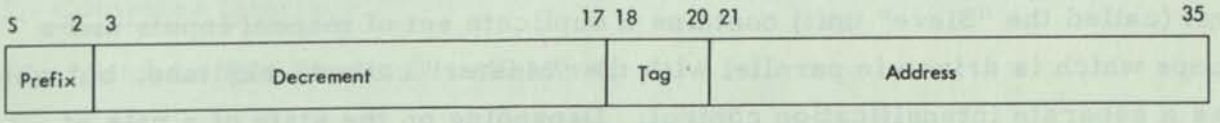
The IBM Selectric Input/Output Writer has been incorporated into the Console for both input and output through the Direct Data Channel. It is wired so that the user types in black and the computer in red. The keyboard layout and character set are those worked out by MITRE Corporation, and known as the MITRE #2.

Two alarm clocks are provided for display timing. These are set by program to ring (i.e., cause direct data interrupt) at any desired interval from 50 microseconds to 128 milliseconds. Other sources of interrupt are: a typewriter key being struck; a push button press; the light pen seeing an active line, point, or character; and from an "edge detect" (a line crossing the edge of the display field or the console field). The "pen see" or "edge detect" interrupts can be individually inhibited by program.

Input facilities provided at each console station are a typewriter, a CTSS Teletype, a light pen, a bank of 9 decimal switches (Digiswitches), two banks of 36 toggle switches, 36 push buttons, three 7-bit shaft encoders and a 3-dimensional rate-control joy stick (crystal ball). The computer can read the settings of these switches and controls whenever it pleases. There is no hardware relation between any of these controls and any Console output function. They are entirely interpreted by the computer program. For example, although it is anticipated the crystal ball (3-D joy stick) will be most effective for controlling rotation, the computer can interpret it any way the programmer desires.

2. Mode of Operation from the Computer

The Display Console interprets 36-bit output words from the Direct Data Channel as commands to it. The Prefix Field of the word (sign bit and bits 1 and 2) categorizes the type of command. The Tag Field (bits 18, 19, and 20) further specifies certain commands. The Address and Decrement Fields, and for certain commands the Tag Field, contain the particular parameters for the command.



Section D contains a detailed description of the action taken upon each command; Tables I and II in the Appendix summarize the command format.

Commands to be transmitted to the Console are arranged in the computer memory in groups of consecutive locations called display lists. Since the data channel can directly access words stored in this manner, the only time the computer's attention is required is when a new list must be started

(or an old list started over). Each command "steals" one memory cycle from the computer.

When the Display Console receives an output word from the Direct Data Channel it responds with a Direct Data Demand (DDD) pulse, interprets the command and commences performing the operation called for (e.g. drawing a specified straight line). The DDD pulse releases the Direct Data Channel to fetch the next output word, i.e., the next command. The Display Console will ignore this new command until it has completed the previous one and the Data Channel will hang up waiting for the DDD pulse. When one command is completed, the Console immediately processes the next awaiting command, etc. Each output word from the Data Channel is looked upon as a new command by the Console except that the Character Generate command puts the Console in a special mode whereby it processes successive output words as characters. This mode of operation, which requires an escape provision, is discussed in detail in the section of Character Generation.

The Console also can input to the Direct Data Channel 36-bit words which can be the setting of an internal register, toggle switches, or other similar source. The ten Output Sense Lines provided by the Direct Data Channel are used for selection of these inputs, and also control other Console functions. The Direct Data Interrupt and the ten Input Sense Lines allow the Console to signal special conditions to the computer. In general the Input Sense Lines identify the source of the Interrupt. Table III in the Appendix depicts the Sense Line bit assignments. Their use is amplified in Section E.

3. Second Operator's Station

A second operator's station has recently been added to the system. This unit (called the "Slave" unit) contains a duplicate set of manual inputs and a scope which is driven in parallel with the "Master" cathode-ray tube, but which has a separate intensification control. Depending on the state of a pair of control flip/flops which are set by a control command, a picture being generated appears on either the Master, the Slave, both or neither. Although called Master and Slave, the stations are in fact equivalent in capability, and two operators can work with different displays by time-sharing the console display generating system.

4. Remote Storage Displays

One of the goals of Project MAC is to provide display capabilities at every terminal. The ESL Display Console is admirably suited to serve as a research tool for experiments aimed at achieving a solution to the remote display problem. A current experiment involves using the Console as a display generator to drive a number of direct-view storage tubes located remotely. Within Technology Square, remote displays can be driven in parallel with the master and slave scopes by sending x and y deflection signals and the intensification signals over coaxial cable. Another experiment involves sending these same signals over three telephone lines equipped with analog data sets.

Two direct-view storage displays will shortly be installed in the public console area at Technology Square for general use. A "snap shot" mode of operation will be used, in which a single copy of a picture will be transmitted and stored on the CRT. Since the writing speed of present storage tubes is not as fast as that of the master and slave scopes, it is necessary to slow the basic operating clock frequency of the Console by a factor of 10 when writing on the storage tubes. A "slow clock" command (bit 10 of the Set C Control Word) has been added to the Console for this purpose. The same control also selects the remote display for intensification. Bits for addressing a particular remote display and for controlling erasure will also be assigned in the Set C Control Word. Special software is also being written to handle the different format required for the remote displays (maximum of 30 characters per line, only special characters can be used, addressing, etc.).

C. SPECIFICATIONS AND PERFORMANCE

1. Computer Interface

Matches IBM Direct Data Channel

36 data bits In

36 data bits Out

10 sense lines In (direct to CPU)

10 sense lines Out (direct from CPU)

1 Channel Interrupt

DEC logic levels (0 and -3 volts) out and in. Requires DEC/7090 level conversion unit (see ESL Memo 9474-M-1).

2. Internal Specifications

Input Power	115 ± 10 volts, 60 cycles, single phase at 18 amps.
Active Scope Size	9 3/8 inches by 9 3/8 inches containing 1024 points by 1024 points
Memory	None (operates from display lists stored in the computer memory, and accessed through the Direct Data Connection)
Clock Rate	Normal 555.55 KC (1.8 μsec between clocks) Slow 69.44 KC (14.4 μsec between clocks)
Line Plotting	
Line Plotting Rate	a point each clock (1.8 μs). This point can be a step of 0, 1, 2, 4, or 8 scope increments in ±h and in ±v (0, 1, 2 for lines to be rotated)
Line Length	0 to 1023 increments in ±x and in ±y without magnification. 0 - 2046 increments by steps of 2 in ±x and ±y with magnification. Thus, Δx and Δy require 10 bits plus sign each.
Random Point Plotting	
Point Plotting Rate	a point every 40 microseconds
Point Plotting Range	2^{13} = 8192 horizontal and vertical positions. Of this only 2^{10} = 1024 will appear on the scope. h = 0, v = 0 is center of screen.
Straza Symbol Generator	
Symbol Code	6 bits to produce one of 64 symbols matching KSR-35 Teletype
Symbol Size	0.1, 0.15, 0.23, or 0.35 inches high
Symbol Plotting Rate	a character every 60 μ sec*

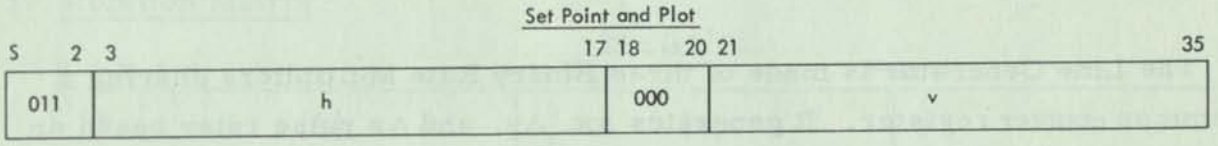
* The Straza Symbol Generator Model 11-64 is capable of producing a character every 10 μsec, but the present magnetic deflection system will not work at this speed. When dual deflection systems (magnetic for position electrostatic for characters) are installed in both the Master and Slave, scopes, the full Straza speed can be used.

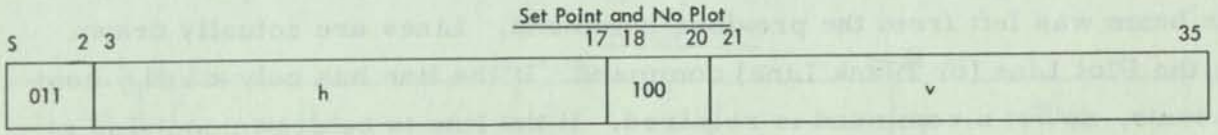
Special Character Generator

Symbol Code	35-bit code to produce any symbol on a 5 x 7 dot matrix
Symbol Size	0.14, 0.28, or 0.56 inches high
Symbol Plotting Rate	a character every 72 μsec
Alarm Clocks	Programmable real time interrupt between 50 microseconds and 6.4 milliseconds by 50 microsecond increments (fast clock), or between 1 millisecond and 128 milliseconds by 1 millisecond increments (slow clock).

D. COMMANDS

1. Point Plotting

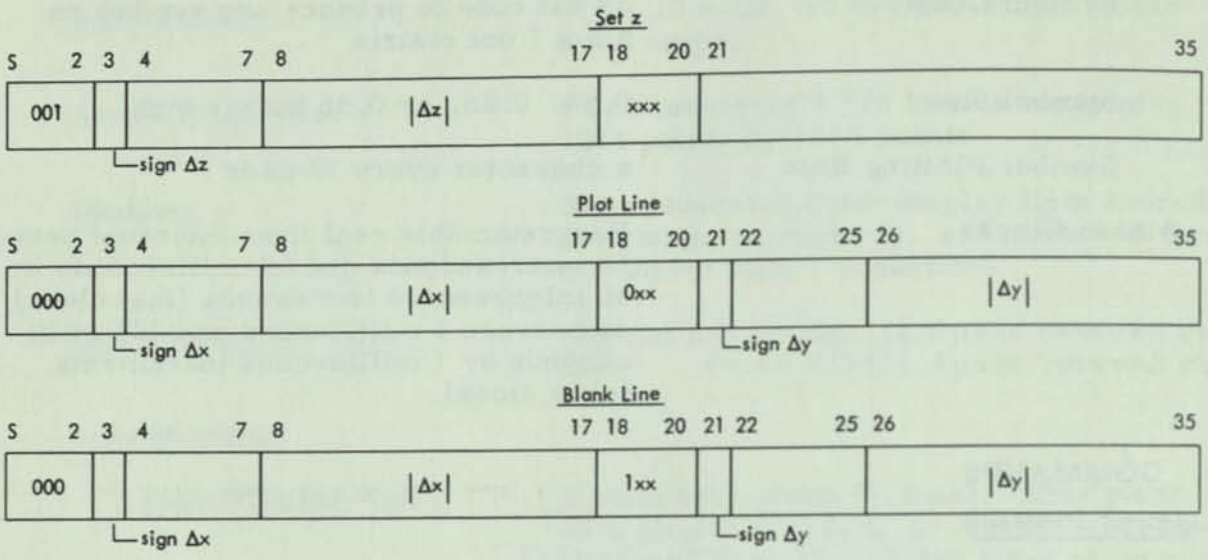




Individual points can be specified and plotted in a manner identical to the DEC PDP Type 30 Scope. This is done by setting h and v (the registers which directly drive the position coils on the crt) and calling for plotting (IBM bit 18 = 0). The electron beam can be positioned without plotting by giving the "Set h, v" command with IBM bit 18 = 1.

The beam can be positioned to any spot in the Total Console Field. As seen in Table II (Appendix), h is specified by the least significant bits of the decrement portion and v by the least significant bits of the address portion of the 36-bit command.

2. Line Generation



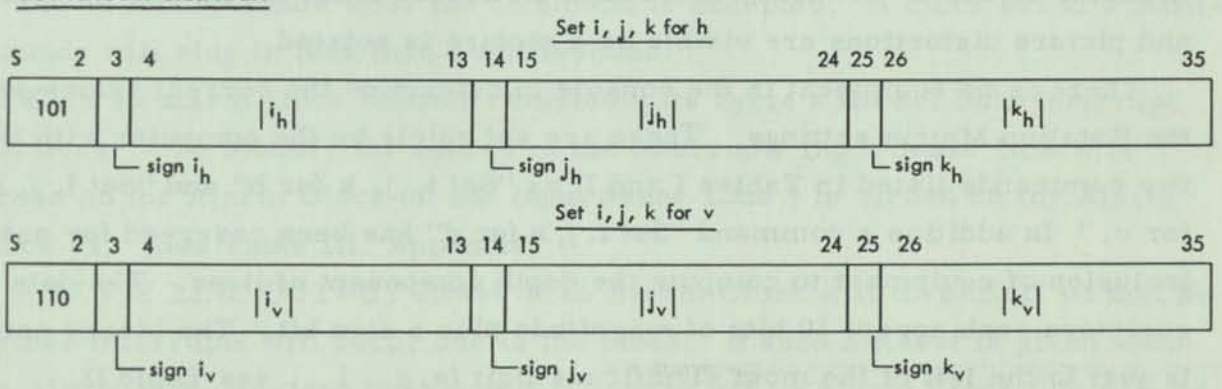
The Line Generator is made of three Binary Rate Multipliers sharing a common counter register. It generates Δx, Δy, and Δz pulse rates based on values set into it by the computer. The line will be begun at whatever point the beam was left from the previous command. Lines are actually drawn by the Plot Line (or Blank Line) command. It the line has only x and y components, no Set z command is required. If the line to be drawn contains a z component, a Set z command must precede the Plot Line (or Blank Line). The second command is necessary even if the line has only a z component ($\Delta x = \Delta y = 0$). Completion of the Plot Line (or Blank Line) command zeros the z register.

The number of x increments desired is placed in the decrement field and the number of y increments in the address portion of the Plot Line command. Z is put into the decrement of the Set z command. All three are sign-magnitude numbers with the sign in the most significant bit of the 15-bit field and the magnitude in the least significant 10 bits. If the line is to be visible, bit 18 of the Plot Line command should be a zero. A blank line is drawn by making bit 18 a one. Blank lines are useful in repositioning the beam without breaking the continuity of a drawing. This point will be clarified in the discussion of the Rotation Matrix. Since the line generators contain 10 bits (plus sign), the longest possible line that can be drawn with normal dot spacing (1/100 inch steps) is 1023 increments (9 3/8 inches) in both x and y,

the diagonal of the Scope Field. Dot spacing can be increased by a C Control command as discussed in a later section.

Lines are not restricted to the Scope Field, but can be drawn on any portion of the Total Console Field. A special programmable interrupt is available to alert the computer when a line is drawn which crosses the edge of the Scope Field (the portion of the Total Console Field which is visible) or when it leaves the Total Console Field entirely. An Edge Detection (ED) flip flop is provided which is set by bit 22 of the C Control command. When ED is ON (bit 22 of the last C Control command was a ONE), lines crossing the Scope Field edge in either direction will cause the interrupt. When ED is off (bit 22 of the last C Control command was a ZERO) the interrupt will occur only if the line leaves the Total Console Field.

3. Rotation Matrix



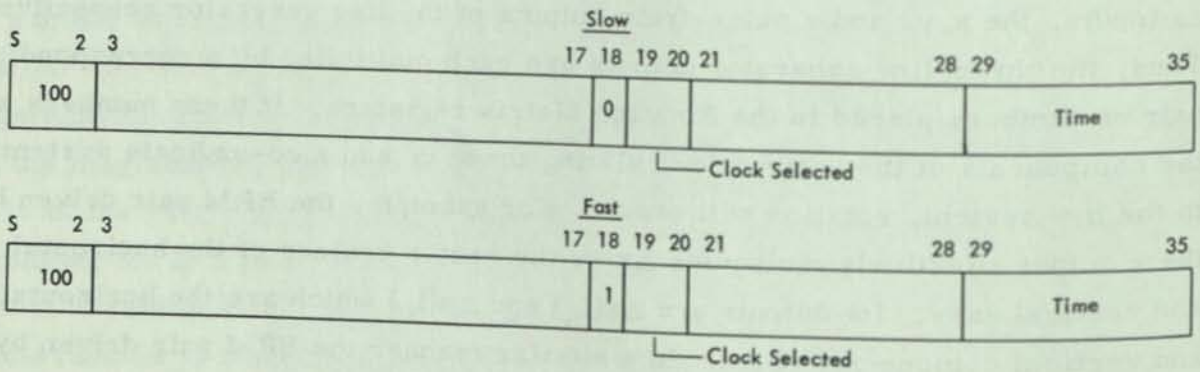
A three-dimensional rotation matrix has been included in the ESL Display Console, which works with the three-dimensional line generator. The rotation matrix units consist of three sets of 10-bit BRM pairs which have as inputs, the x, y, and z pulse-train outputs of the line generator respectively. Thus, the three-line generator outputs are each multiplied by a corresponding pair of numbers placed in the Rotation Matrix registers. If these numbers are the components of the unit vector relating the x, y, and z co-ordinate system to the h, v system, rotation will occur. For example, the BRM pair driven by the x output effectively multiplies Δx by the vector cosines of the horizontal and vertical axes. Its outputs are $\Delta x(i_h)$ and $\Delta x(i_v)$ which are the horizontal and vertical components of Δx. In a similar manner the BRM pair driven by the y output of the line generator provides output $\Delta y(j_h)$ and $\Delta y(j_v)$ the horizontal and vertical components of Δy. The third BRM pair, driven by the z output of the generator, produces $\Delta z(k_h)$ and $\Delta z(k_v)$, the horizontal and vertical components of Δz.

The hardware automatically performs the summing of $\Delta x(i_h)$, $\Delta y(j_h)$ and $\Delta z(k_h)$ into the horizontal incremental input, and the summing of $\Delta x(i_v)$, $\Delta y(j_v)$ and $\Delta z(k_v)$ into the vertical incremental input. Thus, if the proper values are stored into its six data registers ($i_h, i_v, j_h, j_v, k_h, k_v$), the Rotation Matrix will automatically rotate whatever lines are specified in the x, y, and z axes into the horizontal (h) and vertical (v) axes of the scope.

It should be recognized that there is no inherent equipment limitation that says the values placed in the Rotation Matrix data registers must be the proper vector cosines. The only limitation is that the numbers must all be less than ONE. The closest to ONE that can be set into these registers is 1777_8 which represents the number 1023/1024. The nature of the BRM is such that it is the 1024th pulse that will be lost. Since the longest line that can be drawn is 1023 increments, all steps will be plotted and no error will occur. It should be noted, however, that round off errors from the BRM's do exist and picture distortions are visible as a picture is rotated.

There is no equipment in the console to determine the correct values for the Rotation Matrix settings. These are set solely by the computer with the two commands listed in Tables I and II as "Set i, j, k for h" and "Set i, j, k for v." In addition a command "Set i, j, k for d" has been reserved for possible inclusion of equipment to compute the depth component of lines. The data registers each accept 10 bits of magnitude plus a sign bit. The binary point is just to the left of the most significant digit (e.g., i_{v9}, see Table II, Appendix).

4. Alarm Clocks



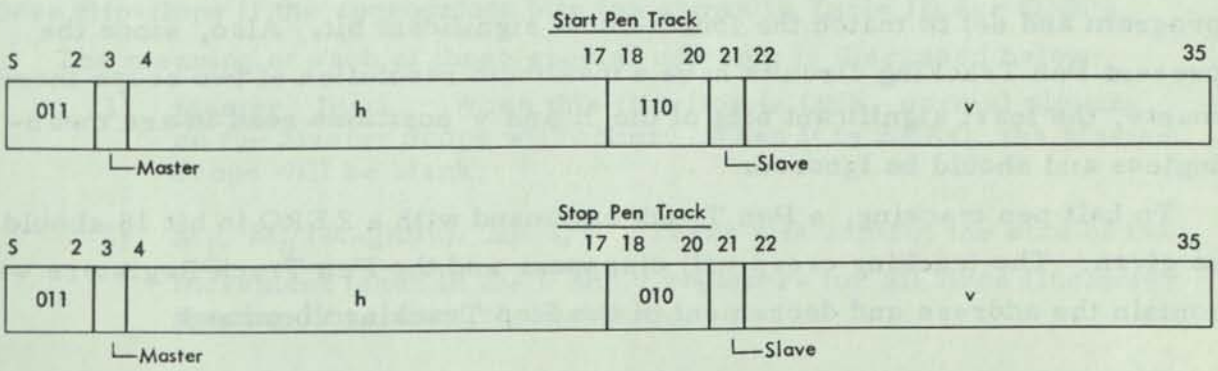
Two programmable Alarm Clocks are provided to give the computer a real-time interrupt source. There is provision in the order code for inclusion of two more Alarm Clocks as specified by bits 19 and 20 of the command. At present only 00 and 01 are active. The clocks are 7-bit counters which are preset by the last seven bits of the address field of the "Set Alarm Clock" command and are driven by either a 50-microsecond or a 1-millisecond clock oscillator. Bit 18 determines which clock source is to be used.

The 50-microsecond clock provides delays of up to 6.4 milliseconds. The 1-millisecond clock provides delays up to 0.128 seconds. Longer delays must be programmed by a succession of shorter ones. Because the clock oscillators are asynchronous, it is impossible to predict the accuracy of the clock closer than one oscillator cycle. Thus, if a clock is set to 67 and the 1-millisecond clock is specified, an interrupt will occur somewhere between 67 and 68 milliseconds after the command is accepted. A clock set to 0 milliseconds will ring in less than 1 millisecond.

When an alarm clock counter completes its cycle a Direct Data Interrupt will occur. To identify the source of the interrupt, Input Sense Line 8 is turned on for Alarm Clock 00 and Input Sense Line 9 is turned on for Alarm Clock 01. (See Table III, Appendix)

A SENSE LINE OUTPUT Reset of an Alarm Clock will turn it off so that no further Interrupts will occur due to the clock. If such a Reset is given while the Alarm is ticking, an interrupt will occur, but no Alarm Clock flag will be set. If the Reset is given when the flag is already up, no interrupt will occur. If an Alarm Clock command is given to an already ticking Alarm, it will reset to the new time. An Interrupt will occur at this time, but no flag will be set.

5. Light Pen Track



In its application in Computer-Aided Design Studies, the ESL Console is used a great deal for tracking the movements of the light pen. To lessen the pen tracking burden on the computer, an automatic tracking feature has been added to the Console, making use of the line generator for generating the tracking cross. There are separate pen-track registers for tracking on the Master and Slave independently.

A Pen Track command to start tracking on the Master (bit 3 = ONE) causes the Master Pen Track Register to be loaded with the horizontal and vertical position specified by the decrement and address fields respectively. The hardware will generate a tracking cross and maintain its up-to-date position automatically every 10 milliseconds. Each Pen Track cross requires about 1 millisecond, thus pen tracking uses 10 percent of display time for one pen.

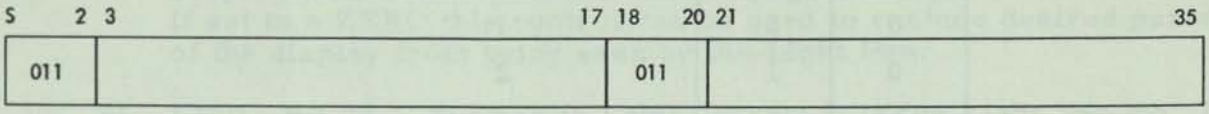
The computer may read the contents of the Pen Track Registers, whenever it desires, in the same manner that it reads the other input information available, as explained in a later paragraph. An important restriction exists however. Reading the Pen Track Register for the Master causes loss of the current h, v register contents, so it is best read after completion of a display list.

Pen Tracking on the slave is initiated by a Pen Track command with bit 21 = ONE. It operates in an identical manner to Master Pen Tracking, except that reading of the Slave Pen Track Register does not cause loss of the h, v registers. If both Master and Slave Pen Tracking generation are taking place, each takes 1 millisecond out of 10, so only 8 milliseconds out of 10 is left for display time.

Since pen tracking cannot occur outside the Scope Field, the Pen Track Registers contain only 10 bits each. Thus, the highest-order three bits of the h and v position read in from Pen Track Registers are extraneous, and are liable to contain false information. In order to interpret the pen position correctly as being on the Scope Field, these bits should be masked by the program and set to match the fourth-most significant bit. Also, since the present Pen Tracking circuits have a maximum resolution of two scope increments, the least significant bits of the h and v positions read in are meaningless and should be ignored.

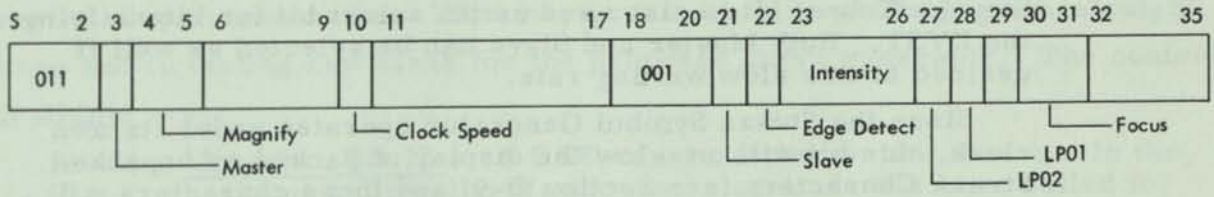
To halt pen tracking, a Pen Track command with a ZERO in bit 18 should be given. The tracking cross will disappear and the Pen Track Registers will contain the address and decrement of the Stop Tracking command.

6. End of File



This command merely sets up a pulse on the End of File Interrupt line into the Direct Data Channel. This is a separate interrupt line from the Direct Data Interrupt, as discussed in IBM Special Features Bulletin for the Direct Data Channel (RPQ M90976). The End of File signal also completes all Data Channel commands. The value of the End of File command is that it provides a way to terminate the Display List processing by the Data Channel without requiring that the number of output words in the list be counted ahead of time. The Address and Decrement fields are ignored by the Console, but can be read by the A-core system following an End-of-File interrupt. The current A-core system* uses this feature to provide several types of transfers useful in subroutining picture data.

7. Set C Control Word



Within the Display Console there are a number of special control flip-flops which affect various aspects of the display. This command will set these flip-flops if the appropriate bits (as shown in Table II) are ONE's.

The meaning of each of these special controls is discussed below:

1) Master. Bit 3. When this flip-flop is ONE, normal plotting on the Master Scope will occur. When it is ZERO, the Master Scope will be blank.

2) M_1, M_0 (Magnify). Bit 4, 5. These bits control the size of the increment taken in the h and v registers for all lines (including spacing lines after characters).

---

* Memorandum MAC-M-201

M_1	M_0	Scope Increments/step (= n/1024 scope width)
0	0	1
0	1	2
1	0	4
1	1	8

The Δx, Δy, Δz of a line designate how many steps to make for that line in each direction. The M bits determine the size of each step. Thus, the length of a displayed line is dependent both on Δx, Δy, Δz and the setting of M_1, M_0. The advantages of using larger increments are that they allow either longer lines to be drawn with a single command, or the same length lines to be drawn faster (a line with Δx, Δy, Δz all halved and M_0 = ONE plots the same line in half as much time).

Restriction. M_1 should be ZERO for all lines which are to be rotated. This is because the rotation matrix accumulates steps from Δx, Δy and Δz at the same clock time, and with M_1 on, it would thus be possible for the rotation matrix to demand a step of 12 or 24 scope increments which the h and v registers cannot do. M_1 is most useful for plotting grid lines for graphs, which can be drawn 4 or 8 times faster but are not expected to rotate.

3) Slow Clock. Bit 10. This bit controls the setting of a flip-flop (SLOW) which controls whether the basic console clock operates at its normal rate (a pulse every 1.8 μsec), or at a slower rate (a pulse every 14.4 μ sec) for driving the remote Direct View Storage Tubes. It is also used as the select bit for intensifying the DVST. Both Master and Slave can be selected as well if desired at this slow writing rate.

Since the Straza Symbol Generator operates under its own clock, this bit will not slow the display of packed or unpacked Straza Characters (see Section D-9) and these characters will plot poorly on remote display. Special characters, however, are satisfactory since their generation rate is controlled by the console clock.

4) Slave. Bit 21. This flip-flop controls the second scope in the same manner as the Master flip-flop. Both can be ON simultaneously if identical displays are desired.

5) ED. Bit 22. This bit sets the Edge Detection flip-flop which determines whether to create a Direct Data Interrupt upon crossing the Scope Field edge (ED = ONE) or upon crossing the Total Console Field edge (ED = ZERO).

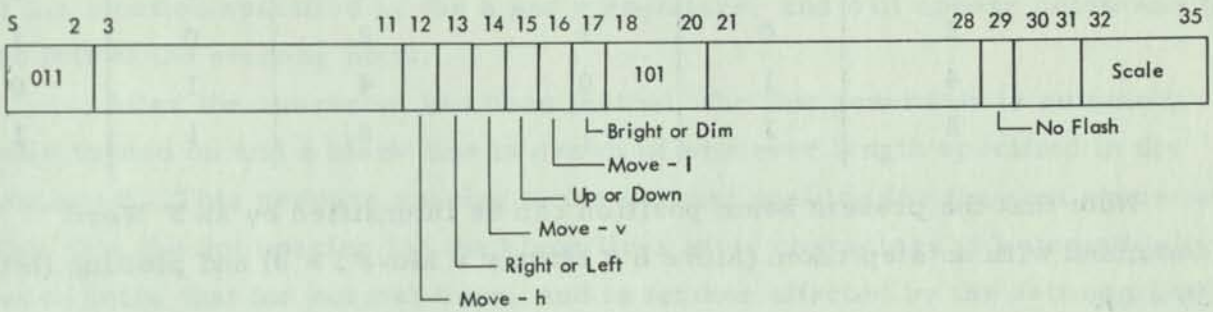
6) I_3 through I_0. Bits 23 - 26. These are the Intensity Control flip-flops. Considering Bit 26 as the least significant bit, larger numbers cause brighter displays.

7) LP02. Bit 27. When set to a ONE this flip-flop enables Light Pen No. 2 (Slave Pen) to respond to points and lines drawn on the scope. It has no effect on Pen Tracking with Light Pen No. 2. If set to a ZERO this control can be used to exclude desired parts of the display from being seen by the Light Pen.

8) LP01. Bit 28. Similar to LP02, except it is for Light Pen No. 1 (Master Pen).

9) F_1, F_2. Bits 30, 31. These flip-flops eventually will affect the focusing of the beam, but are not presently connected.

8. Set F Control Word



The F control word allows the computer access to the increment inputs to the h and v registers directly. In this way the computer can simulate any sort of display generator. This may prove useful in producing non-straight lines and in testing new ideas for the proposed curve generator. The controls available are:

1) Move-h. Bit 12. A ONE causes an increment to occur in the horizontal direction of whatever magnitude has been called for in the Scale h_1, h_0 flip-flops (bits 32, 33 of this command).

2) Right or Left. Bit 13. This control determines the direction of any horizontal step called for by move-h. If a ONE, the step will be taken to the left. A ZERO calls for the step to be to the right.

3) Move v. Bit 14. Similar to Move-h, but in the vertical direction. Its magnitude is controlled by the Scale v_1, v_0 flip-flops (bits 34, 35 of this command).

4) Up or Down. Bit 15. This determines the direction of any vertical step called for by Move-v. A ONE causes steps to be taken downward.

5) Move-I. Bit 16. Similar to Move-h, but it increments or decrements the Intensity Register. The step is always just one unit.

6) Bright or Dim. Bit 17. This determines whether Move-I commands intensify or dim the beam. A ONE in bit 17 causes the beam to get dimmer.

7) No Flash. Bit 29. This controls whether the point called for by this command is to be plotted or not. A ONE inhibits plotting.

8) Scale h_1, h_0, v_1, v_0. Bits 32-35. These flip-flops determine the size of the increments that are to be taken in h and v. The following table applies:

Horizontal Step Size	Bit 32 h_1	Bit 33 h_0	Vertical Step Size	Bit 34 v_1	Bit 35 v_0
1	0	0	1	0	0
2	0	1	2	0	1
4	1	0	4	1	0
8	1	1	8	1	1

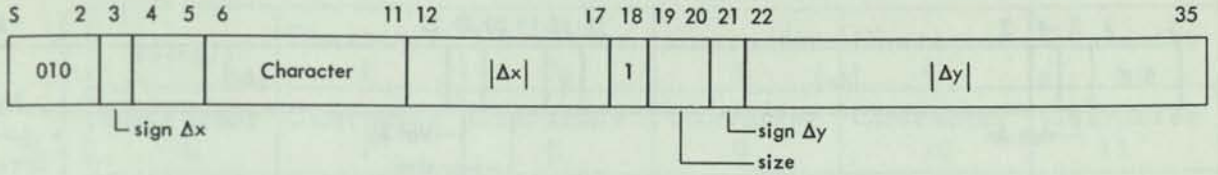
Note that the present beam position can be intensified by an F Word command with no step taken (Move h = Move v = Move I = 0) and plotting (bit 29 = 0).

9. Character Generation

The Display Console contains two symbol generation systems: a Straza Symbol Generator for producing 64 standard symbols, and an M.I.T. constructed Special Character Generator for creating any pattern on a 5 x 7 matrix for display of non-standard symbols. In addition, a flexible automatic format control (spacing between characters) is provided by making use of the line generator. Table VI in the Appendix lists the character code for the Straza Symbol Generator.

There are three modes of operation of the Display Console in generating characters. In the first mode (Unpacked Straza Mode), a single 6-bit character is given with each command. In the Packed Straza Mode, six characters are packed into a single output word. The third mode is the one in which non-standard symbols are produced using a full 36-bit word to specify each character.

a. Unpacked Mode

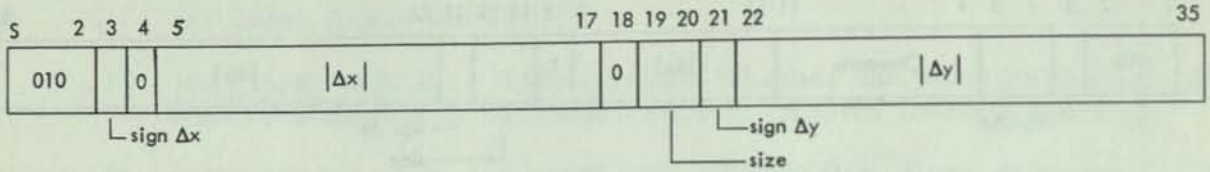


In the unpacked mode, the 6-bit code of the character is held in bits 6 through 11 (see Table II in the Appendix). The Δx spacing after the character is contained in bits 12 through 17, with the sign of Δx in bit 3. The Δy spacing is held in the address portion of the word and is 10 bits in magnitude with sign in bit 21. The character is drawn by the Straza Character Generator starting at the location specified by the h and v registers, and will appear below and to the left of the starting point.

After the character has been plotted, the line generator is automatically turned on and a blank line is drawn of whatever length specified in the command. This permits spacing to the correct position for the next character. Note that the dot spacing for the blank lines after characters is automatically set to twice that for normal lines, and is further affected by the setting of the Magnify (M_1, M_0) flip-flops. Thus, the largest Δx character spacing is 63 increments of 2 each (1.26 inches) with M_1 and M_0 set to ZERO, 63 increments of 4 each (2.52 inches) if M_0 is ONE and M_1 is ZERO, or 63 increments of 8 each (5.04 inches) if M_1 is ONE and M_0 is ZERO.

It is also important to note that both the character generator and line generator outputs bypass the rotation matrix, and thus are unaffected by it. By making the character spacing not rotate it is possible to associate written text with the end point of a line and have the text follow the end point as the line is rotated around on the scope. Note that if a block of several lines of text is attached in this manner, the spacing from the end of one line of text to the start of the next should be done by generating a blank character with appropriate $-\Delta x$, $-\Delta y$ to produce a carriage return which will not be affected by rotation. If a line generate command was used instead, this line would be affected by the rotation matrix and text line 2 would rotate about the end of text line 1.

b. Packed Mode



This mode for character generation was included to allow more dense packing of characters for standard textual output. It is made to conform to the Long Word Routines of the AED Compiler.

The Packed Character Generator command sets the Line Generator to the character spacing desired and puts the Console into a special mode in which it processes succeeding words as blocks of six characters in the format shown below. The first character (bits S through 5) of the first word following the command specifies the number of characters to follow it (i.e., how many characters are contained in the Long Word), and whether or not this is the terminating Long Word. Bits 1 through 5 give the number of characters (up to 30) and bit S designates whether to terminate this special character processing mode (bit S = ZERO) after this Long Word, or to treat the succeeding word as a new Long Word (bit S = ONE).

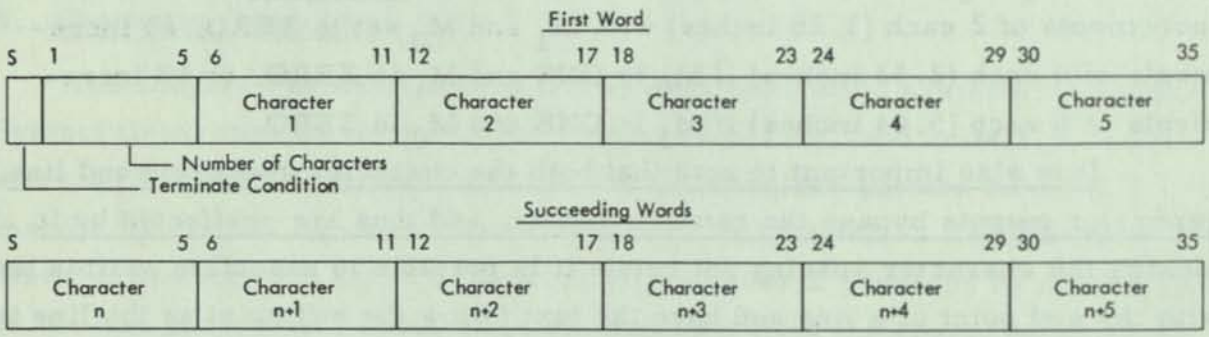


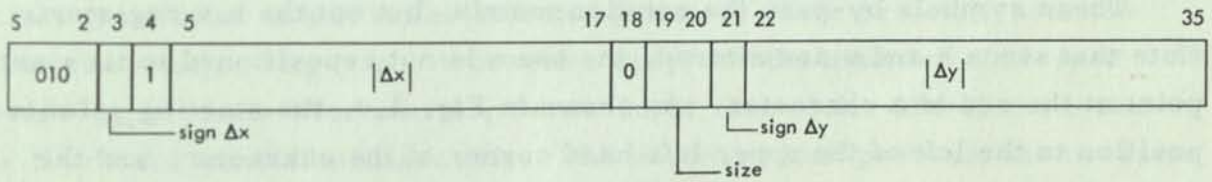
Figure 2.3 illustrates a sample display list for the packed character generate mode. The Console will display the 17 characters of the First Long Word, the one character of the second Long Word and the six characters of the third Long Word in a string, each succeeding character spaced by the amount Δx, Δy as called for in the command.

If the Scope Field Edge is crossed by the line generator during character spacing and the Edge Detect flip-flop (ED) is ON, an Interrupt is generated identical to that described under Line Generation. No such interrupt will occur if the character itself spills over the edge.

	PACKED CHARACTER GENERATE COMMAND					
1st Long Word	110001	Character 1	Character 2	Character 3	Character 4	Character 5
	Character 6	Character 7	Character 8	Character 9	Character 10	Character 11
	Character 12	Character 13	Character 14	Character 15	Character 16	Character 17
2nd Long Word	100001	Character 1				
3rd Long Word	000110	Character 1	Character 2	Character 3	Character 4	Character 5
	Character 6					
	NEXT COMMAND					

Fig. 2.3 Sample Packed-Character Display List

c. Special Character Mode



This mode of character generation permits the creation of any symbol desired on a matrix array of dots. A Special Character Generator command loads the Line Generator with Δx, Δy information for spacing after the array is displayed, then puts the Display Console into a mode in which it processes each succeeding word as 35 bits to be converted into a 5 x 7 array of dots. The 36th bit (bit S) is used to determine whether to terminate the mode (bit S = ZERO) or to process the next word as another array (bit S = ONE). The format of the array is shown in Fig. 2.4.

To intensify any of the points of the array the appropriate bit of the data word is set to a ONE. Dot spacing is normally 2 in this mode, but by setting the SIZ 1 flip-flop ON (Bit 19 = 1), this can be expanded to 4. These can be expanded again by a factor of two by setting the Magnify flip-flop (M_0) ON. M_0 also effects the line drawn after the character, but the

Start → X	6	12	18	24	30
	25	19	13	7	1
	31	2	8	14	20
	15	9	3	32	26
	21	27	33	4	10
	5	34	28	22	16
	11	17	23	29	35
				End →	X

Fig. 2.4 Assignment of Bits in Special Character

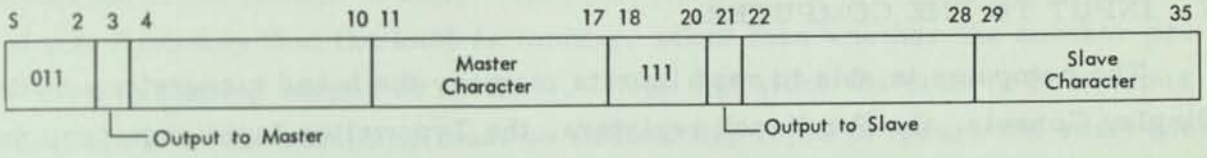
SIZ 1 flip-flop does not. Although 5 x 7 is the standard size in this mode, it is easy to build up larger, more complicated symbols by blocking groups of these arrays together.

These symbols by-pass the rotation matrix, but not the h, v register. Note that since h and v are altered, the beam is not repositioned to its starting point at the end of a character. As shown in Fig. 2.4, the starting point is one position to the left of the upper left-hand corner of the character, and the beam will end one position below the lower right-hand corner. In spacing the beam for the next character, this repositioning must be accounted for. Character spacing is still twice normal size in this mode. The rather strange bit pattern comes about because of the way the matrix is generated, and, because the data register and shifting logic used for the special characters are the same as those used for the packed Straza Characters. To form the matrix, the line generator increments across the top row from left to right, drops down one row and increments from right to left, etc. For each increment, bits 1 through 35 of the data word are rotated left six places and bit position 1 is tested to decide whether to intensify. In making up bit patterns for special characters, it is suggested that a chart such as Fig. 2.4 will be helpful.

10. Typewriters

A pair of Selectric Typewriters using the Mitre No. 2 character set are connected to the console, one for the Master station, the second for the Slave. These provide direct access to the 7094 with keyboard input and allow hard-copy output. The color shift is arranged so that the operator types in black and the computer in red. The 7 bit (odd parity) character appears in the 7 least significant bits of the address field (master typewriter) or decrement field (slave typewriter) along with a flag in the most significant bit. Table VII in the Appendix lists the typewriter code.

a. Output



The Central Processor causes a character to be typed by transmitting the above command to the console. This command loads the appropriate typewriter output buffer selected (Master, Slave or both), sets the typewriter output flag (i.e., typewriter busy) for the machine addressed (Sense Input Line 4 for Master, Sense Input Line 5 for Slave) and locks the Keyboard for 500 milliseconds. The keyboard lock prevents the operator from typing in while output is occurring. The typewriter flag remains ON until the machine concerned is ready to accept a new character. This is generally before the mechanical cycle is complete and can therefore range from 30 to 130 milliseconds. The maximum output typing rate, however, is 15 characters per second.

b. Input

When an operator strikes a key on either typewriter, the character is loaded into an input buffer, the appropriate flag (Input Sense Line 4 for Master, 5 for Slave) is set, the keyboard is locked, and an interrupt sent to the Central Processor. The CPU reads the character in the same manner as other inputs, i.e., the Sense Line Output code to select the typewriters (see Table IV) is presented and the word read in. The character and flag appear in the same position in the word on input as on output (Table V).

Reading the character into the computer unlocks the keyboard. At computer speeds, this short lock time is unnoticeable. Note that there is a single typewriter flag for each machine, and that it is used both for input and output. Thus the Central Processor must remember if it has initiated typewriter output when interpreting the flag.

c. Echo Checking

Echo checking can be done for each character output by performing the read sequence before outputting the succeeding character. The character that the typewriter actually types is automatically set into the read buffer at the time the typewriter output flag is reset.

E. INPUT TO THE COMPUTER

The computer is able to read into its memory the h and v registers of the Display Console, the Pen Track registers, the Typewriter Input register, and the data from the various devices on the control panel, which are described in the next section. To accomplish this, the computer first selects the input source by a Present Sense Lines Instruction (PSL) with the last 4 Sense Lines coded to the desired unit. Table IV indicates the existing input sources and their codes for selection. Table V indicates the bit positions of the input words from h, v registers, the shaft encoders and crystal ball, the Pen Track registers and the typewriters. The push buttons and switches are labeled as to their bit assignments. Selecting an input source causes the Console to put the data on the input lines to the Direct Data Channel. The computer must then Read Select the Direct Data Channel to input a single word.

Due to some special hardware considerations on the push buttons the programmer should be sure that he does not leave the Data Channel Read Selected for a long period of time (1 millisecond or more). Doing so causes any push buttons held down to be read several times (i.e., interpreted as a sequence of several pushes of the same button).

Read Control of the Console has been made independent of Write Control. It is thus possible to read information from the console into the computer while the console is hung in some output condition, such as an Edge Crossing. This permits the CPU to read h, v to determine the exact location of the crossing and then release the Console by means of a Reset Flags to finish the display list.

Identical facilities are provided at the master and slave stations, with separate selection codes for read in to the computer; thus the following discussions apply either to the master or the slave stations.

1. Control Panel

Figure 2.5 illustrates the control panel provided for input to the computer. At the bottom of the panel are the three 7-bit binary Shaft Encoders. There is no internal relation between the position of these knobs and anything else in the Display Console, and their interpretation is handled purely by program. There is no interrupt caused when these are moved, and if the program is concerned with the position of these knobs, it must sample their contents often enough to make sense out of the data. There is no indication of the direction that the knob is turning, other than whether the number presented is getting larger or smaller. Note that since the code is continuous (modulo 2^7), the sampling must be sufficiently rapid to insure the shaft does not rotate more than 180° between samples. Normally 30 times per second is a satisfactory sampling rate.

To the right of the knobs are located two switches marked BEAM OFF and POWER and a button marked RESET. The BEAM OFF switch originally blanked the beam so the operator could protect the screen from being burned by an improper program (one that continuously intensifies the same spot). It was found that by keeping the intensity level adjusted properly this switch function was not necessary so it was disconnected. Since then, the switch has been wired to override the no-plot control, to allow display of all lines, blank or plotted, on screen or off. This has proved useful for debugging purposes. The RESET button resets the active control flip-flops of the Console. In this state the Console is ready to process a new command. The computer is not alerted in any way when this button has been activated. The POWER switch is disconnected on the Master but is active on the Slave Control Panel.

Behind the panel on the Master, a Sonalert buzzer has been wired to the WHOA flip-flop to give the computer an active alarm. The WHOA flip-flop was chosen because in the time sharing system it usually indicates a computer hang-up condition. The function of this flip flop is described later.

Above the knobs are two banks of 36 toggle switches, each of which constitutes one input word. These switches provide arbitrary control functions. There is no internal relation between the position of these switches and anything

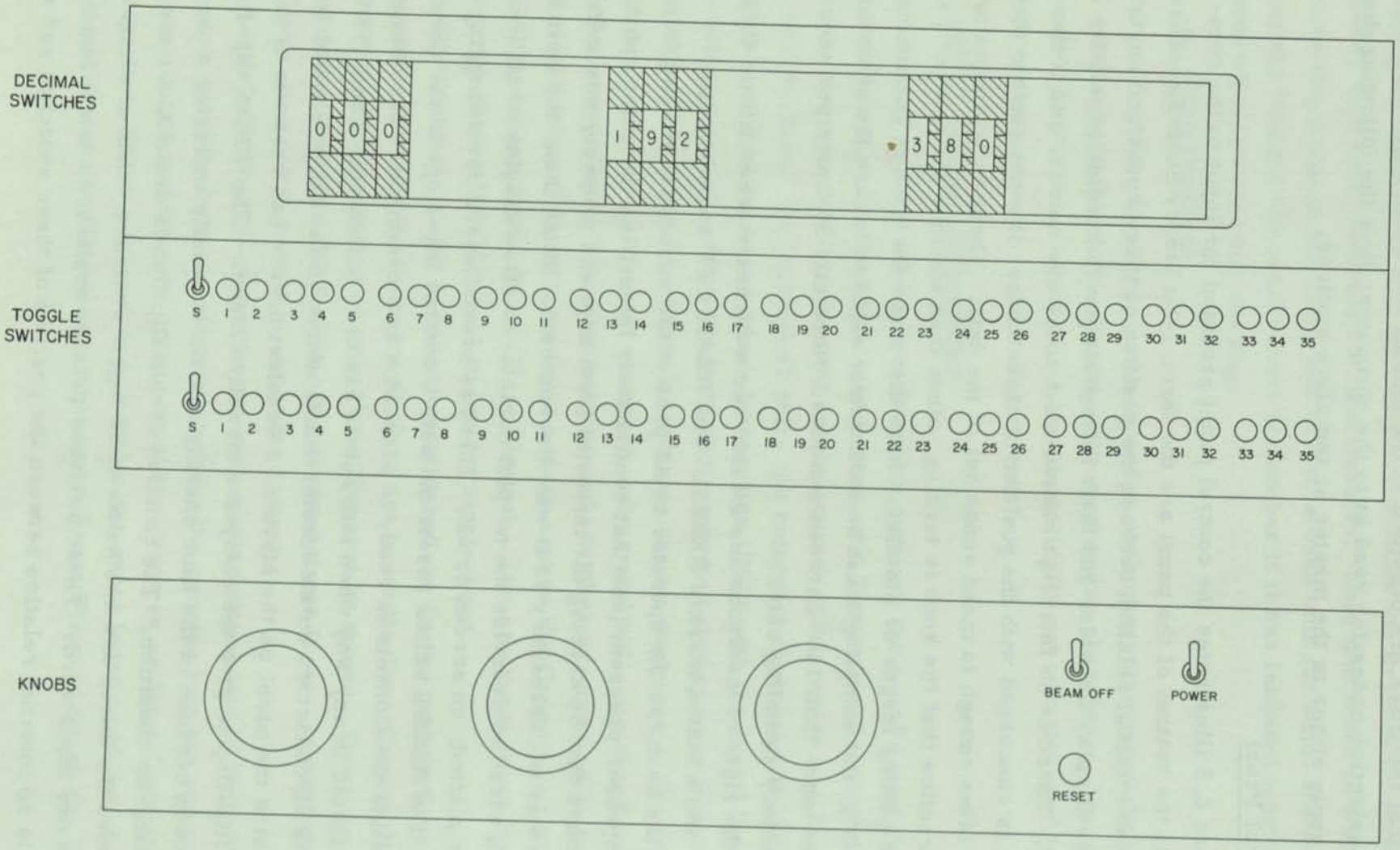


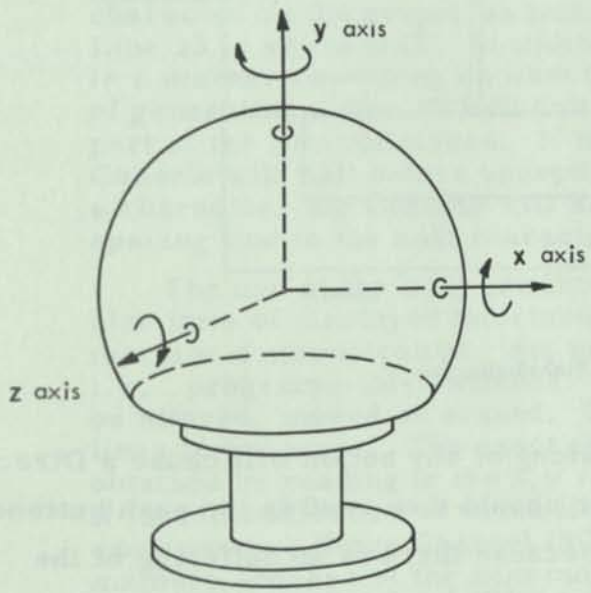
Fig. 2.5 Control Panel

else in the Display Console, and their interpretation is handled purely by program. Like the knobs, these switches are sampled by the computer at whatever rate is set by the program.

Above the toggle switches is a bank of 9 binary coded decimal switches. These are mechanically mounted on horizontal slides in such a way that they can be spaced into any grouping. Thus, they can be interpreted as nine individual 10-position switches or three 1000-position inputs or whatever arrangement the user desires. The bit assignment of the switches is shown in Table V.

2. Crystal Ball

The Crystal Ball or "Globe," illustrated in Fig. 2.6, has spring-loaded



FOR EACH AXIS	
CODE $g_3\ g_2\ g_1\ g_0$	MEANING*
1 0 0 0	Highest negative rate
1 1 0 0	Middle negative rate
0 1 0 0	Lowest negative rate
0 0 0 0	Reset Position. Zero rate
0 0 1 0	Lowest positive rate
0 0 1 1	Middle positive rate
0 0 0 1	Highest positive rate

* Considering clockwise rotation as positive

Fig. 2.6 Crystal Ball

limited rotation about three axes of rotation, and provides input codes to the computer for 3 positions in each direction about each axis. The device is intended to be used as a three-step rate control to provide a natural easy control over the rotation of a three-dimensional object viewed on the screen. The input codes are merely processed through the Console, however, and do not directly effect any console registers. Thus the Crystal Ball can be used in any way that the programmer desires. Fig. 2.6 shows the relation between

ball position and binary input for each axis (assuming positive rates are for clockwise rotation).

3. Push Buttons

The Push Button Box (Fig. 2.7) is a unit containing 36 push buttons for

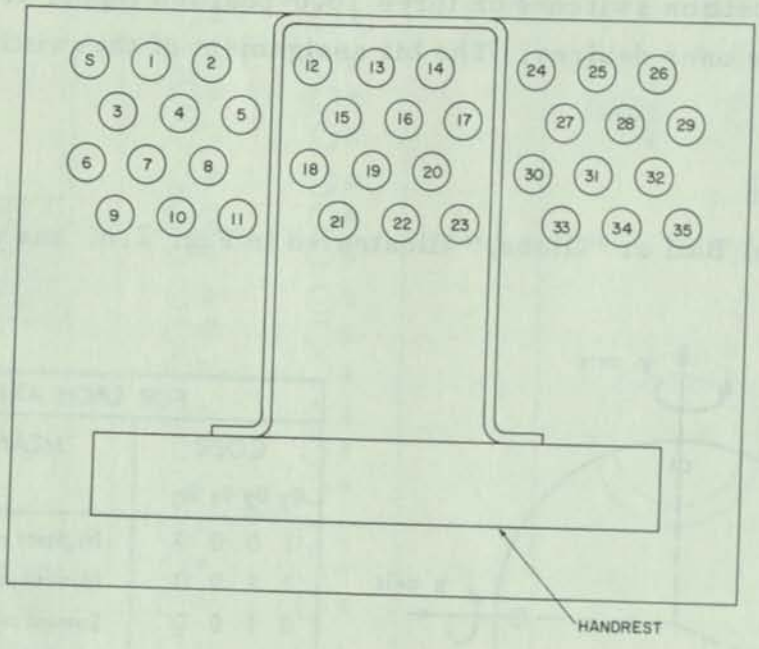


Fig. 2.7 Push-Button Box

control of the computer program. Pushing of any button will cause a Direct Data Interrupt to occur. The computer should then read in the push buttons to determine which one was pushed. Because there is no buffering of the push button data, it is possible (though not likely) for the computer not to get around to sample the buttons before a button is released. It is, therefore, advisable to give some visible indication on the display that a push button has been recognized. Like the switches and shaft encoders, no wired control has been assigned to any buttons. Their interpretation is controlled strictly by the program.

The Push Button box has been designed for easy use without looking at it, so that the operator's attention is not diverted from the CRT Display. The buttons are arranged like a keyboard and they are designed for very light pressure. A special hand rest is provided with spacing bars separating

groups of buttons to aid in locating oneself without looking at the box. A plastic holder is also provided above the box to slip an identification card into, so each user can associate the buttons with the particular uses in his program.

4. Interrupts

The Direct Data Interrupt is used to call the Central Processor into play when special conditions occur in the Console. The sources of interrupt are as follows:

1) Alarm Clocks. One of the alarm clocks has rung. Input Sense Lines 1 and 2 identify these interrupts.

2) Typewriters. A typewriter key has been struck. Input Sense Lines 4 and 5 identify these interrupts.

3) Light Pen. If the light pen is enabled and it sees a point, line or character on the scope, an interrupt is generated and Input Sense Line 13 is set to ONE. In addition, the Display Console is "hung up" in a manner depending on what it was doing. If it is in the process of generating a line, it will halt as soon as the pen responds, leaving part of the line unfinished. If the pen responds to a Set Point, the Console will halt before accepting the next command. If the pen sees a character, the Console will halt just before plotting the blank spacing line to the next character.

   The use of the light pen interrupt to identify to the CPU a particular item of displayed information is an important means of man-machine communication. Set points can be used as "light buttons," i.e., programmable switches. Displayed lines or characters can be altered, moved, or erased. The Central Processor can identify lines in two ways. The exact point on a line seen by the pen can be obtained by reading in the h, v register when this interrupt occurs. If just identification of which line has been seen is enough, the CPU can execute a Store Channel (SCH) instruction which will give the memory address of the next output command. Note that the only way to identify which of six packed characters was seen is to establish each character location and compare it to the h, v register.

   Presenting Sense Lines (PSL) with the Reset Flag bit ON (bit 13) will release the line generator and allow an interrupted line to be completed. If the h, v register word is read in by the computer, bits 3 and 21 contain flags which idenfity which light pen (Master or Slave) saw the line. Bit 3 flag is set by the Master Pen. Bit 21 flag is set by the Slave Pen. Both are reset by Reset Flags Sense Line Output (bit 13).

4) Edge of Scope. This interrupt, which is identified by bit 14 of the Sense Input Word, occurs if the Edge Detect flip-flop (ED) is ON and the Scope Field edge is crossed by a line during Line Generation or spacing for Character Generation. If ED is OFF, it occurs when a line is drawn completely off the Total Console Field. When this interrupt occurs, the Line Generator is hung up in the same way as described in the previous paragraph.

The same two options described for the Light Pen interrupt are available to the programmer after the Edge of Scope interrupt. Thus, the CPU can find the exact edge co-ordinates of the line crossing, or just identify which line is the culprit.

5) Push Buttons. When any of the 36 push buttons available for input to the computer are pushed, an interrupt is caused and bit 15 of the Sense Input Word is turned ON. The CPU can then input these buttons to determine the particular button pushed. The sense line flag is reset by a Reset Flags Sense Line Output (bit 13).

F. INTERRUPT PROCESSING

The programmer should be aware that the processing of Direct Data Interrupts is fraught with hazards. Some are imposed by the nature of the 7094 Direct Data Channel, others by the characteristics of the Display Console.

One such difficulty arises with the automatic disabling of interrupts when an Interrupt Trap is taken. If the CPU performs a WRS or RDS to the Direct Data, the interrupt is automatically reenabled. Disaster then strikes if a second interrupt occurs while the CPU is still in the interrupt subroutine since the main program return address (core location 00003) will be wiped out.

The most successful procedure has been to Enable with ZERO upon entering an interrupt subroutine, process the interrupt, and then test the Sense Lines to see if any other interrupts have occurred before returning. This can be done because each interrupt source has its own flag.

Halting the Data Channel in the midst of a display list to process an interrupt can also cause difficulties. For example, it is necessary for the CPU to know reliably where to restart the list. If the computer merely does a Store Channel (SCH) to preserve the list address, and then a Reset Channel (RCH) to start the list again, it cannot be sure that the Display Console did not accept the command that was being held on the output lines just before the RCH. If this occurs, an address is skipped which produces annoying jumps in the display. To avoid this problem the WHOA flip-flop was added to the Console which causes the Console to hang up before accepting the next command. The CPU sets the WHOA flip-flop with a Present Sense Lines (PSL) instruction with bit 11 of the Sense Output word = ONE. By giving such an instruction several cycle times before an SCH, the CPU can be sure that the Data Channel address will not be stored just as it is about to change state.

If the Data Channel is stopped on a data word for packed or special characters, the Console is in a condition where it expects the next output word to be a data word, not a command. If the CPU were to attempt to output any command at this time, it would be improperly interpreted. For a command to be processed correctly in this situation, the Console must be reset with a Panic Reset before the command is put out. This, however, clears the Console's memory that it is awaiting a character word. Restarting on the character word would foul up the rest of that one pass through the display. The computer must either restart the display back at the beginning or it must be able to identify the stored channel address as being that of a Character Word and restart at the last Set Point Command. Note that since reading the Console does not effect its state, the above restriction does not apply if the CPU only wants to read conditions of the Console upon interrupt.

There do exist quite innocent ways to hang-up the computer with the Console if the programmer is not careful. For example, if the CPU attempts a Write or Read Select when the Data Channel is still processing, the CPU will halt on the Select instruction waiting for the Data Channel disconnect. If the console halts on an Edge Detect or Light Pen See during this time, the Interrupt will not get to the CPU. This can be avoided by performing a TCOD* instruction before the Select instruction.

A similar hang-up can occur if the Direct Data Interrupts are disabled (as they are in an Interrupt subroutine) and the CPU is in a TCOD* loop. If the Console stops on a Light Pen Seen during this time, the Data Channel is halted and the TCOD* loop is never broken because the interrupt is not enabled. To avoid this, it is best to Reset the Channel (RCH) on any interrupt and TCOD in a loop through a program that examines the Sense Lines for any added interrupt flags, stores them, and Resets Flags.

Another difficulty has been encountered when a display list is broken up into blocks in the data channel and the programmer tries to identify Light Pen See Interrupts with specific words of the display list. For example, if the last item on a display list block transmitted by an IOCP is a line, the data channel will have moved on to the next data block (next IOCP), which can be anywhere in memory, before the line is even well started. If a Light Pen See Interrupt occurs on this line, its correct channel location will not be one less than the Channel Location Address as expected. The easiest way to

insure that this does not happen is to add some NO-OP command to the Console (ZERO length line, ZERO Z wd or ZERO F word with No Plot ON) at the end of each block.

APPENDIX

COMMAND FORMAT, BIT ASSIGNMENTS AND CHARACTER CODES

Table I

COMMAND FORMAT FOR ESL DISPLAY CONSOLE

7094 Bits							Function	Option
Prefix			Tag					
S	1	2	18	19	20	4		
0	0	1	X	X	X		Line Generation	Set Z
0	0	0	0	X	X		" "	Plot
			1	X	X		" "	Blank
0	1	0	0	S_1	S_0	0	Character Generation	Packed Mode
			0	X	S_0	1	" "	Special Symbol
			1	S_1	S_0		" "	Unpacked Mode
0	1	1	0	0	0		Set Point	h, v Plot
			1	0	0		" "	h, v Blank
			0	0	1		Load	Control Word C
			1	0	1		"	Control Word F
			Y	1	0		Light Pen Track	Master or Slave
			0	1	1		End of File	
			1	1	1		Typewriter Output	Master or Slave
1	0	0	0	A_1	A_0		Set Alarm Clock	Slow
			1	A_1	A_0		" " "	Fast
1	0	1					Set k, j, k, for h	
1	1	0					Set i, j, k, for v	
1	1	1					Set i, j, k, for d (Depth coordinate. Not installed yet)*	

Notes: S_1 and S_0 specify character size.

A_1 and A_0 specify which of four Alarm Clocks, (only two installed).

X indicates "don't care".

Y specifies START (1) or STOP (0)

* This command is presently used to address the Calcomp Plotter System. See "Graphical Output Device," A. Rutchka, Memorandum MAC-M-210, December 21, 1964.

Tab

COMMAND BIT ASSIGNMENTS

7094 Bits	Set h, v	Set z	Generate Line	Unpacked Character Generator	Packed Character Generator	Special Character Generator	Set i, j, k for h	Set i, j, k for
S	0	0	0	0	0	0	1	1
1	1	0	0	1	1	1	0	1
2	1	1	X	0	0	0	1	0
3		Sign z	Sign x	Sign x	Sign x	Sign x	Sign i_h	Sign i_v
4					0	1	i_{h9}	i_{v9}
5	h_{12}						i_{h8}	i_{v8}
6	h_{11}			c_5			i_{h7}	i_{v7}
7	h_{10}			c_4			i_{h6}	i_{v6}
8	h_9	z_9	x_9	c_3 } Char.	x_9	x_9	i_{h5}	i_{v5}
9	h_8	z_8	x_8	c_2	x_8	x_8	i_{h4}	i_{v4}
10	h_7	z_7	x_7	c_1	x_7	x_7	i_{h3}	i_{v3}
11	h_6	z_6	x_6	c_0	x_6	x_6	i_{h2}	i_{v2}
12	h_5	z_5	x_5	x_5	x_5	x_5	i_{h1}	i_{v1}
13	h_4	z_4	x_4	x_4	x_4	x_4	i_{h0}	i_{v0}
14	h_3	z_3	x_3	x_3	x_3	x_3	Sign j_h	Sign j_v
15	h_2	z_2	x_2	x_2	x_2	x_2	j_{h9}	j_{v9}
16	h_1	z_1	x_1	x_1	x_1	x_1	j_{h8}	j_{v8}
17	h_0	z_0	x_0	x_0	x_0	x_0	j_{h7}	j_{v7}
18	Plot (0) or Blank (1)	X	Plot (0) or Blank (1)	1	0	0	j_{h6}	j_{v6}
19	0	X	X	S_1 } Size	S_1 } Size	X	j_{h5}	j_{v5}
20	0	X	X	S_0	S_0	S_0 (Size)	j_{h4}	j_{v4}
21			Sign y	Sign y	Sign y	Sign y	j_{h3}	j_{v3}
22							j_{h2}	j_{v2}
23	v_{12}						j_{h1}	j_{v1}
24	v_{11}						j_{h0}	j_{v0}
25	v_{10}						Sign k_h	Sign k_v
26	v_9		y_9	y_9	y_9	y_9	k_{h9}	k_{v9}
27	v_8		y_8	y_8	y_8	y_8	k_{h8}	k_{v8}
28	v_7		y_7	y_7	y_7	y_7	k_{h7}	k_{v7}
29	v_6		y_6	y_6	y_6	y_6	k_{h6}	k_{v6}
30	v_5		y_5	y_5	y_5	y_5	k_{h5}	k_{v5}
31	v_4		y_4	y_4	y_4	y_4	k_{h4}	k_{v4}
32	v_3		y_3	y_3	y_3	y_3	k_{h3}	k_{v3}
33	v_2		y_2	y_2	y_2	y_2	k_{h2}	k_{v2}
34	v_1		y_1	y_1	y_1	y_1	k_{h1}	k_{v1}
35	v_0		y_0	y_0	y_0	y_0	k_{h0}	k_{v0}

Note: X in bits S-2 and 18-20 indicates "don't care".

v	Set C Control	Set F Control	Light Pen Track	Typewriter Output	Alarm Clock	EOF	7094 Bits
	0	0	0	0	1	0	S
	1	1	1	1	0	1	1
	1	1	1	1	0	1	2
	Master (1)		Master (1)	Master (1)			3
	M_1 } Increment Size						4
	M_0 }						5
							6
							7
			h_9				8
			h_8				9
	Clock } Normal (0) Slow (1)		h_7				10
			h_6	Parity			11
		Move h	h_5	c_5			12
		Right (0) Left (1)	h_4	c_4			13
		Move v	h_3	c_3 } Master Char.			14
		Up (0) Down (1)	h_2	c_2			15
		Move I	h_1	c_1			16
		Bright (0) Dim (1)	*	c_0			17
	0	1	On (1) Off (0)	1	Fast (1) Slow (0)	0	18
	0	0	1	1	A_1 } Clock	1	19
	1	1	0	1	A_0 }	1	20
	Slave (1)		Slave (1)	Slave (1)			21
	ED (Edge Detect)						22
	I_3 } Intensity						23
	I_2 }						24
	I_1 }						25
	I_0 }		v_9				26
	LP02 (Slave)		v_8				27
	LP01 (Master)		v_7				28
		No Flash	v_6	Parity	t_6		29
	F_1 } Focus		v_5	c_5	t_5		30
	F_2 }		v_4	c_4	t_4		31
	S_{h1} } Scale		v_3	c_3 } Slave Char.	t_3 } Time		32
	S_{h0} }		v_2	c_2	t_2		33
	S_{v1} }		v_1	c_1	t_1		34
	S_{v0} }		*	c_0	t_0		35

* See note on Table V.

Table III
SENSE INPUT AND OUTPUT WORDS

Sense Input Number	IBM Bit	Meaning
1	8	Alarm Clock No. 1 Rung
2	9	Alarm Clock No. 2 Rung
3	10	Push Button 2 (Slave)
4	11	Master Typewriter Flag
5	12	Slave Typewriter Flag
6	13	Light Pen Seen
7	14	Edge of Scope
8	15	Push Button 1 (Master)
9	16	Used by Calcomp Plotter System*
10	17	Used by Calcomp Plotter System*

Sense Output Number	IBM Bit	Meaning	
1	8	Reset Alarm Clock No. 1	
2	9	Reset Alarm Clock No. 2	
3	10	Used by Calcomp Plotter System*	
4	11	WHOA	
5	12	Panic Reset	
6	13	Reset Flags	
7	14	Select Register Bit 0	See Table IV
8	15	Select Register Bit 1	
9	16	Select Register Bit 2	
10	17	Select Register Bit 3	

* "Graphical Output Device," A. Rutchka, Memorandum MAC-M-210, December 21, 1964.

Table IV

SENSE LINE OUTPUT BITS FOR INPUT SELECTION

IBM Bit				Device Selected
14	15	16	17	
0	0	0	0	h,v Register
0	0	0	1	Unassigned
0	0	1	0	Push Buttons 1
0	0	1	1	Push Buttons 2
0	1	0	0	Auto Pen Track Register 1
0	1	0	1	Auto Pen Track Register 2
0	1	1	0	Switch Bank 1 B (lower bank)
0	1	1	1	Switch Bank 2 B (lower bank)
1	0	0	0	Shaft Encoders and Crystal Ball 1
1	0	0	1	Shaft Encoders and Crystal Ball 2
1	0	1	0	Switch Bank 1 A (upper bank)
1	0	1	1	Switch Bank 2 A (upper bank)
1	1	0	0	Typewriter 1
1	1	0	1	Typewriter 2
1	1	1	0	Decimal Switches 1
1	1	1	1	Decimal Switches 2

* Note: Master = 1, Slave = 2

Table V

DATA INPUT BIT ASSIGNMENTS

7094 Bits	h, v Registers	Shaft Encoders		Decimal Digiswitches		Light Pen 1 Register	Light Pen 2 Register	Typewriter	
S				$d9_3$	9th (leftmost) Digiswitch				
1				$d9_2$					
2				$d9_1$					
3	LP1S	g_{x3}		$d9_0$				RD(M) Read Flag Master	
4		g_{x2}	Crystal Ball x axis	$d8_3$	8th				
5	h_{12}	g_{x1}		$d8_2$					
6	h_{11}	g_{x0}		$d8_1$					
7	h_{10}	g_{y3}		$d8_0$					
8	h_9	g_{y2}	Crystal Ball y axis	$d7_3$	7th	PT1 h_9	PT2 h_9		
9	h_8	g_{y1}		$d7_2$		PT1 h_8	PT2 h_8		
10	h_7	g_{y0}		$d7_1$		PT1 h_7	PT2 h_7		
11	h_6	g_{z3}		$d7_0$		PT1 h_6	PT2 h_6	Parity	Master Char.
12	h_5	g_{z2}	Crystal Ball z axis	$d6_3$	6th	PT1 h_5	PT2 h_5	c_5	
13	h_4	g_{z1}		$d6_2$		PT1 h_4	PT2 h_4	c_4	
14	h_3	g_{z0}		$d6_1$		PT1 h_3	PT2 h_3	c_3	
15	h_2	$e_{\ell 6}$		$d6_0$		PT1 h_2	PT2 h_2	c_2	
16	h_1	$e_{\ell 5}$		$d5_3$		PT1 h_1	PT2 h_1	c_1	
17	h_0	$e_{\ell 4}$		$d5_2$	5th	*	*	c_0	
18		$e_{\ell 3}$	left encoder	$d5_1$					
19		$e_{\ell 2}$		$d5_0$					
20		$e_{\ell 1}$		$d4_3$					
21	LP2S	$e_{\ell 0}$		$d4_2$	4th			RD(S) Read Flag Slave	
22		e_{c6}		$d4_1$					
23	v_{12}	e_{c5}		$d4_0$					
24	v_{11}	e_{c4}		$d3_3$					
25	v_{10}	e_{c3}	center encoder	$d3_2$	3rd				
26	v_9	e_{c2}		$d3_1$		PT1 v_9	PT2 v_9		
27	v_8	e_{c1}		$d3_0$		PT1 v_8	PT2 v_8		
28	v_7	e_{c0}		$d3_3$		PT1 v_7	PT2 v_7		
29	v_6	e_{r6}		$d2_2$	2nd	PT1 v_6	PT2 v_6	Parity	Slave Char.
30	v_5	e_{r5}		$d2_1$		PT1 v_5	PT2 v_5	c_5	
31	v_4	e_{r4}		$d2_0$		PT1 v_4	PT2 v_4	c_4	
32	v_3	e_{r3}	right encoder	$d1_3$		PT1 v_3	PT2 v_3	c_3	
33	v_2	e_{r2}		$d1_2$	1st (rightmost) Digiswitch	PT1 v_2	PT2 v_2	c_2	
34	v_1	e_{r1}		$d1_1$		PT1 v_1	PT2 v_1	c_1	
35	v_0	e_{r0}		$d1_0$		*	*	c_0	

* The present Pen-tracking circuits are such that the smallest step is equal to two scope increments. These bits are reserved for PT h_0 and PT v_0 if the smallest tracking step is changed to one scope increment.

TABLE VI STRAZA SYMBØL GENERATØR CØDE ASSIGNMENT

ØCTAL CØDE	SYMBØL
00	0
01	1
02	2
03	3
04	4
05	5
06	6
07	7
10	8
11	9
12	~
13	=
14	'
15	—
16	@
17	&
20	+
21	A
22	B
23	C
24	D
25	E
26	F
27	G

ØCTAL CØDE	SYMBØL
30	H
31	I
32	;
33	.
34	)
35	:
36	↑
37	"
40	-
41	J
42	K
43	L
44	M
45	N
46	Ø
47	P
50	Q
51	R
52	%
53	$
54	*
55	←
56	?
57	!

ØCTAL CØDE	SYMBØL
60	(SPACE)
61	/
62	S
63	T
64	U
65	V
66	W
67	X
70	Y
71	Z
72	>
73	,
74	(
75	]
76	[
77	<

Table VII

IBM SELECTRIC TYPEWRITER CODE

Octal Code	Lower Case	Upper Case
00	0	\
01	1	|
02	2	$
03	3	`
04	4	'
05	5	<
06	6	>
07	7	↑
10	Carriage Return	Carriage Return
11	Index	Index
12	Tab	Tab
13	Space	Space
14	9	/
15	8	↓
16	=	?
17	[0]	[\]
20	)	]
21	☆	←
22	s	S
23	t	T
24	u	U
25	v	V
26	w	W
27	x	X
30	Back Space	Back Space
31	Upper Case	Upper Case
32	Lower Case	Lower Case
33	Keyboard Lock	Keyboard Lock
34	z	Z
35	y	Y
36	,	;
37	[)]	[]]

Octal Code	Lower Case	Upper Case
40	-	_
41	j	J
42	k	K
43	l	L
44	m	M
45	n	N
46	o	O
47	p	P
50	[Carriage Return]	[Carriage Return] *
51	[Index]	[Index]
52	[Tab]	[Tab]
53	[Space]	[Space]
54	r	R
55	q	Q
56	(	[
57	[-]	[_]
60	+	→
61	a	A
62	b	B
63	c	C
64	d	D
65	e	E
66	f	F
67	g	G
70	[Back Space]	[Back Space]
71	[Upper Case]	[Upper Case]
72	[Lower Case]	[Lower Case]
73	[Keyboard Lock]	[Keyboard Lock]
74	i	I
75	h	H
76	.	:
77	[+]	[→]

* Bracketed symbols will print this character on WRITE but are not used on READ