EVANS & SUTHERLAND COMPUTER CORPORATION

CLIPPING DIVIDER

MAINTENANCE MANUAL

February 17, 1970

Evans & Sutherland Computer Corporation
Three Research Road
Salt Lake City, Utah 84112

CLIPPING DIVIDER MAINTENANCE MANUAL

Table of Contents

INTRODUCTION

1.1 Objective of this Manual

This maintenance manual was written for the purpose of providing technical information about the design and function of the Clipping Divider. The manual deals with how the operations of the Clipping Divider (clipper) are implemented. The major portion of the manual deals with the logical design of the clipper. Because the clipper operates at standard TTL logic levels and because the components used are, for the most part, conventional IC's, detailed electrical descriptions are included only when unusual characteristics are involved.

A familiarity with the conception and operation of the Line Drawing System and a working knowledge of the functioning of digital devices is assumed.

1.2 References

For a general introduction to the LDS system, the following should be read:

1. "A Clipping Divider" by Robert Sproull and Ivan Sutherland.
2. "LDS-1 System Reference Manual"

In addition to these documents which provide general information, wirelists, etc are referenced throughout the manual which provide specific information. These documents comprize the real documentation for the system and, in part, this manual is a guide to help in understanding the specific documentation.

It is also this specific documentation which is to be considered more up-to-date and accurate in cases of inconsistancy between these documents and the maintenance manuals.

1.3 Organization of the Manual

The manual is divided and numbered by sections. A few notes on numbering and nomenclature conventions make up Section 2. Section 3 contains a detailed description of the algorithm of the Clipper. Section 4 deals with the Back Panel and relates various functions to physical locations. Sections 5-18 contain information about the individual cards; and Section 19 is an Appendix containing algorithm definitions and read only memory descriptions and wire lists.

We have attempted to keep the organization of the different sections consistant. A few remarks of introduction describe the general objective of the item being discussed and relate this part to the whole. Pertinent references are then given, followed by a discussion of the important details.

It is important, especially in the case of the logic diagrams which are imperative to the understanding of the sections dealing with the cards, that the references be studied at least concurrently with the reading of the manual.

NUMBERING AND NOMENCLATURE CONVENTIONS

2.1 Numbering

Three groups of numbers are especially significant for the documentation package. The first group identifies the parts, subassemblies, and assemblies used to build the system. The second group refers to physical locations on the back panel or on the component boards. The numbers used to identify different sections of the manuals make up the third group.

2.1.1 E & S Part Numbers

Each part used in the system has been assigned an E & S Part Number. These part numbers are related to part descriptions and vendor numbers by the E & S Guzzinta Chart. With one exception, these numbers have no significance other than to identify the actual part used.

The exception is in the case of the E & S part numbers assigned to integrated circuits. These numbers are six-digit numbers with a three-digit dash number appended. The first three digits are 807; the fourth indicates the number of pins on the IC involved; the fifth and the sixth identify the type of IC (2-wide AND-OR-Invert, etc.). The three-digit dash number relates to tolerances in speed, input load, and drive factor. These numbers allow the substitution of different IC's provided that they have the same logical function and pin numbering and meet the tolerance established by the dash number. These numbers decode as follows:

807NID-SLD where

N = Number of pins

2 = 24-28 pins
3 = 36 pins
4 = 14 pins
6 = 16 pins

ID = Identification number attached to the IC (usually relates to a vendor number)

S = Speed (odd numbers indicate fixed speed; even numbers indicate minimum speed allowable)

L = Input load in ma

D = Drive factor in ma/10

S		L	D
0-1	Open	0 = 0.0	0 = 6.0
2-3	74HTTL (5-7ns)	1 = 0.4	1 = 5.4
4-5	Signetics (8-10ns)	2 = 0.8	2 = 4.8
6-7	Medium 74TTL (12-14ns)	3 = 1.2	3 = 2.8
		4 = 1.6	4 = 2.4
		5 = 2.0	5 = 2.0
		6 = 2.4	6 = 1.6
		7 = 2.8	7 = 1.2
		8 = 3.2	8 = 0.8
		9 = 3.6	9 = 0.4

2.1.2 Location Numbers

Back panel positions are identified by card slot numbers and connector numbers. The card slot numbers are shown in Figure 4.1.

Test points and connectors (pins for intercard connections) have been numbered as labeled on the boards. Test points run from 1-18 on the solder side of the board and have letter codes from A to V (not continuous) on the component side. Test points on the component side are labeled by letter on the logic drawings and etchings, but identified by numbers on the wirelist. The correspondance is as follows:

A = 19	H = 25	P = 31
B = 20	J = 26	R = 32
C = 21	K = 27	S = 33
D = 22	L = 28	T = 34
E = 23	M = 29	U = 35
F = 24	N = 30	V = 36

The connectors are numbered from 1-98. The even numbers are on the solder side of the board and the odd on the component side. Connectors 1 and 97 carry Vcc and 2 and 98 carry ground.

The boards used contain 70 bug (IC) positions that are numbered from 10-79. The tens figure identifies the column and the ones figure identifies the row of the bug position.

2.1.3 Manual Numbering

This manual is numbered by sections. Page numbers for each section as well as subsection numbers and figure numbers are prefaced by the number of the section.

2.2 Notations

An effort has been made to keep the nomenclature and notations as consistant as possible throughout the manual. However, a few of the conventions used and not used require a little explanation.

2.2.1 Philosophy of Wire and Card Naming

The names attached to the individual wires and the different cards of the system are somewhat arbitrary, but an attempt has been made to make the name some sort of mnemonic indicating the purpose of the wire or board in question.

Wires on the back panel have names that are not necessarily the same as the name of the signals on the cards. Also, a signal may have one name on one card and another on a different card. This often necessitates the use of the backpanel wirelist when tracing a signal from card to card.

Signals which require a low pulse to enable the desired function, or data signals which are in compliment form, are denoted by the "not" symbols. In the manual and throughout the wire lists, these "not" signals are denoted by an *. On the logic diagrams, the symbol used may be either an ∿ or a superscripted line (e.g., $\overline{\text{CLRAIC}}$).

2.2.2 Capitalization

Wire names are given in upper case letters. If a data bus or a group of signals is referenced, only the bus name or the common part of the signal name is given. This was done to avoid awkward repetitions. For example, the signals *ZINT (0) - *ZINT (16) are referred to simply as the *ZINT bus, and COUNT (Q1), COUNT (Q2), COUNT (Q3), COUNT (Q4) are simply referred to as COUNT.

When a reference to a signal's function is made, the name of the function may not be capitalized even though it corresponds in part with the signal's name. Thus, signals controlling "carry ins" for an adder are sometimes called "carry in signals" and not CARRY IN signals, even though the actual name of the signals may be CARRY IN A, CARRY IN B, etc.

Card names are capitalized as proper names in contrast to the complete capitalization of the wire names. Thus, the "DONE" signal comes from the "Done card."

CLIPPING DIVIDER ALGORITHM

3.1 Understanding the Algorithm

The basic description of the Clipping Divider algorithm is contained in "A Clipping Divider" by Sproull and Sutherland. The following discussion of the algorithm assumes the information in this paper. Several additional documents are essential for the understanding of both the following description and the algorithm itself. The flow diagram of the algorithm (E & S block diagram No. 101123-950) should be studied thoroughly and available for constant reference during the reading of the algorithm description. The algorithm is defined in PDP-9 macros and associated Read Only Memory bits which are included in the appendix. These documents constitute the most accurate definition of the algorithm. Reference to the block diagram showing the registers and busses of the Clipping Divider (E & S block diagram No. 101123-900) will also prove essential.

3.2 Reference

E & S block diagram No. 101123-950, Clipping Divider Algorithm, Clipping Divider.

E & S block diagram No. 101123-900, Registers and Busses.

"A Clipping Divider" by Sproull and Sutherland.

3.3 General Outline

The Clipping Divider (clipper) is basically an arithmetic device. A directive and data are fetched from external devices or from the storage registers of the clipper. The incoming data is put in proper form and loaded into the accumulator according to the directive. If lines are indicated, these lines are compared to a specified window and clipped to determine the point of intersection with the window. In certain cases, the clipped line is put back into page coordinates and sent back to computer memory. The clipped lines are then compared to the viewport and scaled (divided). The results are sent to computer memory and displayed. The appropriate part of the displayed data remains in the storage registers of the clipper for use in the next iteration of the algorithm.

The accumulators of the clipping divider consist of "working" registers and associated adders. Data is manipulated within the accumulator, between the accumulator and storage registers, or to an output device, according to the directions of the control logic. Eight dispatches and 62 steps make up the algorithm. Each step corresponds to a word of Read Only Memory (ROM). The enabled word of Read Only Memory activates a group of bits which control the manipulation of data. A program counter, which is conditioned by the directive received and the state of the data in the accumulators, controls the progression of the clipper through the algorithm.

The following details relate to the individual steps of the algorithm. For the purposes of this discussion the algorithm will be divided into the following steps:

1. Initia;ization, directive and data fetching.
2. Data set up.
3. Register loading.
4. Window set up.
5. Clipping.
6. Page coordinates to memory.
7. Division.
8. Output.
9. Boxing.

A general description of each function and the relation of the individual steps to that function will preceed the detailed description of the steps of the algorithm. The description of the steps will include a few remarks about the function of the step. The numbers attached to the individual steps correspond to the actual ROM word lines activated. The steps are described in the normal sequence in which they occur or in relation to a special function. For these reasons, the steps are not always in numerical order.

After this general description, the ROM bits engaged will be given in three categories.

The ROM bits are divided into ten groups as to function:

1. ROUNDC (R): 6 bits of rounding and injection control.
2. ENABLC (E): 4 bits of control to the "out code" used in clipping, division, and boxing.

3. STOREC (S): 8 bits of control to the storage registers.

4. LROMC (L): 12 bits, 3 each for four 8 x 16 ROM addresses which control the accumulator loading and unloading.

5. STEPC (ST): 9 bits marking specific steps.

6. FLOPC (F): 8 bits of flip-flop control.

7. INOUTC (I): 13 bits of input and output flag control.

8. POKEC (P): 13 bits of poke control data.

9. JUMPA (JA): 8 bits of jump address.

10. JUMPC (JC): 18 bits of jump control.

The information given about the individual bits activated by any step will be given in short descriptive phrases that identify the actual ROM bits. The first seven groups will be delineated under the title "ROM bits". Short descriptions of the bits engaged are given for each of the groups affected. ROM bit groups are identified by letter codes. Bits engaged in group 8 (P) are described under the special heading "Pokes". Groups 9 and 10 have to do with the program counter and are described under the heading "PC". These bit function descriptions of all groups are related to the actual ROM bit wire names and positions by the tables in the appendix.

In addition to the ROM bits, the contents of the working register of the accumulator AFTER the step in question has been performed will be given in diagram from. The four rows and four columns relate to the working register of the clipper and are arranged in the same order as shown in the block diagram showing the registers and busses of the clipper (101123-900). There are four groups of working registers in the clipper as shown in figure 3.1 below. Each group consists of an upper and lower register and an associated adder. The groups are labelled P (for previous) and N (for new) in both X and Y coordinates. This allows the representation of two end points both with two coordinate representations in X and two in Y (These may include Z in the three-dimensional case). The contents of the working registers will be shown in diagram form for each step. The diagrams show the contents after that step has been performed. When especially relevant, data paths to or from the accumulator, to the storage register, or to output devices will be indicated.

WORKING REGISTER LAYOUT

PUAX	PUBX	PUAY	PUBY
PLAX	PLBX	PLAY	PLBY
NUAX	NUBX	NUAY	NUBY
NLAX	NLBX	NLAY	NLBY

U = Uppers A = odds, left
L = Lowers B = evens, right
P = Previous
N = New

STORAGE REGISTER LAYOUT

RAX RBX RAY RBY RAX = R1 RBX = R2 RAY = R3 RBY = R4

Figure 3.1

3.3.1 Initialization, Directive and Data Fetching

The first step in the algorithm provides the possibility of clearing the clipper. Just below (after) this step is the point to which the algorithm returns after each pass. An incoming directive is awaited and acknowledged upon arrival. Incoming data is then accepted (if there is any). If a "fetch" type operation is indicated by the directive, data is fetched from the right side of the storage registers of the clipper, the left side, or both. After the fetching, the algorithm returns to the beginning to await a new directive. If no "fetch" is indicated by the directive, control is transferred to dispatch 1. Dispatch 1 has three possible paths. If more than one input transfer is required to get all of the data into the clipper, the algorithm enters either a "self data" state (data from the INSTANCE register are written over the incoming data) or the "more" state. In the "more" state, Additional input data are awaited and acknowledged upon arrival. In the case that neither of these two steps are called for by the directive, or after these steps have been performed, the clipper goes to dispatch 2 which puts the data in proper form for later operations.

(0) CLEA

This first step clears all of the input/output flip-flops and conditions used by the algorithm. CLEA is entered only when the master reset condition is forced by the reset switch on the clipper control panel or by the master reset level from the main computer.

ROM BITS:	POKES:	PC:
F Clear CUR Clear AGREE Clear AIC	Arbitrary (no pokes and no inhibits)	Jump WAIT

(2) WAIT

Each time the algorithm is finished, it returns to the WAIT state to await a new incoming directive. WAIT transfers the directive into the directive register of the clipper.

ROM BITS:		POKES:	PC:
(ST)	Input 1 (waiting for input and directive)	Arbitrary	PC +1
(F)	Clear CUR		
(I)	Poke directive Input wait		

ACCUMULATOR CONTENTS:

X	X	X	X
X	X	X	X
X	X	X	X
X	X	X	X

X = no change

(3) DATA

The DATA step loads the new data (if any) into the accumulators of the clipper according to the directive. Old data from the SAVE registers are also loaded into the accumulators by the DATA step. The DATA step acknowledges the input of data.

ROM BITS:		POKES:	PC:
(S)	Register true output to AC Register address = 0 (SAVE)	Arbitrary	Dispatch 1 (MORE) unless test 204 (fetch) then jump FETC
(L)	Lower switch = 2 (select joint) Upper switch = 1 (select input)		
(ST)	Input step (special clearing allowed by X and Y)		
(I)	Acknowledge input flag		

ACCUMULATOR CONTENTS: (dependent on directive)

FROM ABS				FROM REL				TO ABS			
D2	D1	D4	D3	D2	D1	D4	D3	0	0	0	0
0	0	0	0	S1	S2	S3	S4	S1	S2	S3	S4
0	0	0	0	0	0	0	0	D2	D1	D4	D3
S1	S2	S3	S4	S1	S2	S3	S4	0	0	0	0

TO REL				SIZE OR SELF			
0	0	0	0	D2	D1	D4	D3
S1	S2	S3	S4	S1	S2	S3	S4
D2	D1	D4	D3	D2	D1	D4	D3
S1	S2	S3	S4	S1	S2	S3	S4

D = incoming data
S = data from
SAVE register

(4) FETC

If the directive indicates that data are to be fetched from the storage registers of the clipper, the FETC state is entered. This step enters the contents of the register selected by the directive into the accumulators. From the FETC state, the clipper enters either the FETL or FETR state.

ROM BITS:	POKES:	PC:
S Use directive address for read Register true output to AC L Upper switch = 2 (select joint) Lower switch = 2 (select joint)	A11	Jump to FETR unless test 54 (odd or 72 bit) then PC+1

ACCUMULATOR CONTENTS:

R2	R1	R4	R3
R1	R2	R3	R4
R2	R1	R4	R3
R1	R2	R3	R4

R = selected register

R1 R2 R3 R4 -- loaded

(5) FETL

In the FETL state, the clipper outputs the left portion (A side) of the storage registers. The memory output flag is raised and the clipper waits with data on the lines for acknowledgement.

ROM BITS:	POKES:	PC:
L Sum switch = 2 (select uppers) Output switch = 1 (PNPN) I Raise outflag for memory data Output wait	ROM address 2 (uppers)	Jump WAIT unless test 50 (72 bit REG) then PC+1

ACCUMULATOR CONTENTS:

R1	R2	R3	R4	
R1	R2	R3	R4	
R1	R2	R2	R2	
R1	R2	R3	R4	
R1		R3		-- output from uppers

(6) FETR

In the FETR state, the clipper outputs the right portion (B side) of the data in the storage registers (e.g., the right and top parts of the WINDOW or VIEWPORT or the XZ and YZ parts of the SAVE registers). Again the Clipping Divider raises the memory output flag and waits for it to be acknowledged.

ROM BITS:	POKES:	PC:
L Output switch = 1 (PNPN) Sum switch = 1 (select lower) I Raiseoutflag for memory data Output wait	Arbitrary	Jump WAIT

ACCUMULATOR CONTENTS:

R2	R1	R4	R3	
R1	R2	R3	R4	
R2	R1	R4	R3	
R1	R2	R3	R4	
R2		R4		-- output from lowers

(7) MORE

In the MORE state, the clipper waits for more incoming data which is written into one half of the accumulators according to the control of the directive. The data accepted in the MORE state may be ZZ data in the 3D case or XY data if data are being retrieved from the stack. The MORE state awaits incoming data and acknowledges it on arrival.

ROM BITS:	POKES:	PC:
S Enable true register output to AC Register address = 0 (SAVE) L Lower switch = 2 (select joint) Upper switch = 1 (select input) I Input wait Acknowledge input flag	Inhibit except more data	Dispatch 2 (SIZR)

ACCUMULATOR CONTENTS

Same as DATA

(10) SELF

Data from the INSTANCE register are written over either the incoming X data or the incoming Y data when the clipper is in the SELF state. Either the X or Y data are thus ignored in this state.

ROM BITS:	POKES:	PC:
S Register true output to AC Register address = 3 (INSTANCE) L Upper switch = 2 (select joint)	Inhibit except self data Clear PU ROM address = 2 (poke uppers)	Dispatch 2 (SIZR)

ACCUMULATOR CONTENTS:

0	0	0	0	
X	X	X	X	I = Instance
I2	I1	I4	I3	X = No change
X	X	X	X	

3.3.2 Data Set Up

After data has been taken into the clipper, the form of the data is corrected by the steps immediately after dispatch 2. If "center size" data was specified, the SIZR (size relative) or SIZA (size absolute) state is used to fix the form of the data correctly. If the plotting of a dot is called for, the DOTT state is entered. If load or box is called for, or after the center size correction steps have been completed, the ENDN state is entered. If line or setpoint is indicated by the directive, or after the DOTT step has been completed, the ENDR state is entered. The ENDN and ENDR steps put the endpoints of the lines into the accumulators. Because it relates to loading or boxing or center size data, ENDN does not put the new end of the line into the SAVE register whereas ENDR does. ENDR initiates dispatch 3. ENDN goes either to dispatch 3 or to the LOAD step for loading the registers of the clipper if loading is called for by the directive.

(11) SIZR

SIZR compliments the new data and loads the lower registers with SAVE register data, but leaves the contents of the SAVE register intact. The data is then relative to the SAVE register data.

ROM BITS:	POKES:	PC:
R Inject P even (next pulse)	ROM address 5	Jump ENDN
Inject P odd (next pulse	(P)	
E Register true output to AC		
Register address = 0 (SAVE)		
L Upper switch = 4 (select compliment)		
Lower switch = 2 (select joint)		

ACCUMULATOR CONTENTS:

-D	-D	-D	-D
S1	S2	S3	S4
D	D	D	D
S1	S2	S3	S4

(12) SIZA

SIZA clears the lower registers of both the N and P sections of the clipper and compliments the P data. The data thus become positive and negative displacements from the origin of the coordinates.

ROM BITS:	POKES:	PC:
R Inject P even (next pulse) Inject P odd (next pulse) L Upper switch = 4 (select compliment)	Clear PL,NL ROM address 5 (poke P)	ENDN Jump

ACCUMULATOR CONTENTS:

-D	-D	-D	-D
0	0	0	0
D	D	D	D
0	0	0	0

(13) DOTT

The step for plotting dots is called DOTT. DOTT replaces the data of the previous end of the line with the data from the new end of the line. DOTT thus changes the data to appear as if it represented a line of zero length.

ROM BITS:	POKES:	PC:
L Sum switch = 3 (select full sum) Lower switch = 1 (select other sum)	Clear PU ROM address 5 (poke P)	PC+1

ACCUMULATOR CONTENTS:

0	0	0	0
N1	N2	N3	N4
X	X	X	X
X	X	X	X

N = N (new) data
X = no change

(14) ENDR

The ENDR step adds the contents of the upper and lower registers, which may be the new data and the contents of the save register or the new data and 0 (in the size absolute case). The data in the N accumulator is then written into the SAVE register for later use.

ROM BITS:	POKES:	PC:
S Write Register address = 0 (SAVE) L Sum switch = 3 (select full sum) Output switch = 2 (NNNN)	All	Dispatch 3 (CURV)

ACCUMULATOR CONTENTS:

P1	P2	P3	P4	
P1	P2	P3	P4	
N1	N2	N3	N4	
N1	N2	N3	N4	
N1	N2	N3	N4	- - Output to SAVE register

(15) ENDN

The function of ENDN is identical to that of ENDR, with the exception that ENDN does not replace the value of the SAVE register. ENDN is used for loading functions or for the start of boxing, or if center size data is specified. In these cases, it is not appropriate to change the SAVE register contents.

ROM BITS:	POKES:	PC:
L Sum switch = 3 (select full sum)	All	Dispatch 3 (CURVE) unless test 310 (box or load) then LOAD

ACCUMULATOR CONTENTS:

P1	P2	P3	P4
P1	P2	P3	P4
N1	N2	N3	N4
N1	N2	N3	N4

3.3.3 Register Loading

If "load" or "box" is called for by the directive, the clipper goes into the LOAD step. This step is preliminary to boxing and has no effect on the loading functions themselves. If boxing is called for, the clipper will go into the boxing sequence (described in section 3.3.9). If loading is called for, the LOAD step exits to dispatch 4 which dispatches to one of four loading steps according to the directive. Each of the four loading functions returns to the WAIT step when finished, except LOADR (load rectangle) which goes to an additional step to test for area in common (AIC) and then to the WAIT step.

(16) LOAD

The LOAD step is a preliminary to boxing. Some of the data in the working registers of the accumulator are replaced with data from the INSTANCE register.

ROM BITS:	POKES:	PC:
S Register true output to AC Register address = 3 (INSTANCE) L Upper switch = 3 (select old and new data) Sum switch = 1 (select lower) Lower switch = 6 (select old and new data) F Clear AGREE	All	Dispatch 4 unless test 110 (BOX)

ACCUMULATOR CONTENTS:

P2	I1	P4	I3
I1	P1	I3	P3
I2	N1	I4	N3
N2	I2	N4	I4

I = INSTANCE register

(20) LOAE

LOAE loads the even (B side) storage registers addressed by the directive with new data. Data from a storage register indicated by the directive address is loaded into the accumulator. The storage registers are then reloaded. New data is written into the even side and old data is reloaded into the odd side.

ROM BITS:	POKES:	PC:
S Use directive address for read Use directive address for write Write Register true output to AC L Sum switch = 6 (select new data) Output switch = 6 (NRNR)	Arbitrary	Jump WAIT

ACCUMULATOR CONTENTS:

P1	P2	P3	P4
P1	P2	P3	P4
N1	N2	N3	N4
N1	N2	N3	N4

(21) LOAO

LOAO loads the odd portion of the storage registers (A side) with new data from the accumulator. Data from the storage registers are loaded into the accumulator. The odd storage registers are loaded with new data, while the old data is reloaded into the evens.

	ROM BITS:	POKES:	PC:
S	Use directive address for read	Arbitrary	Jump WAIT
	Use directive address for write		
	Write		
	Register true output to AC		
L	Sum switch = 6 (select new data)		
	Output switch = 4 (RNRN)		

ACCUMULATOR CONTENTS:

P1	P2	P3	P4
P1	P2	P3	P4
N1	N2	N3	N4
N1	N2	N3	N4

(22) LOAA

If both the even and odd (A and B) portions of the storage registers are to be loaded with new data, the LOAA step is entered.

	ROM BITS:	POKES:	PC:
S	Use directive address for write	Arbitrary	Jump WAIT
	Write		
L	Sum switch = 6 (select new data)		
	Output switch = 2 (NNNN)		

ACCUMULATOR CONTENTS:

P1	P2	P3	P4
P1	P2	P3	P4
N1	N2	N3	N4
N1	N2	N3	N4

(23) LOAR

This step is entered when a 2D specification has called for loading an entire 4-component register. In this case, the new and previous ends of the "line" are treated as the upper right and lower left parameters of a rectangle. Data from the WINDOW registers are loaded into the lower registers of the accumulator in preparation for the next step.

ROM BITS:	POKES:	PC:
R Inject P even (next pulse)	Arbitrary	PC+1
Inject P odd (next pulse)		
Inject N even (next pulse)		
Inject N odd (next pulse)		
S Use directive address for write		
Write		
Register complement output to AC		
Register address = 2 (WINDOW)		
L Sum switch = 6 (select new data)		
Lower switch = 2 (select joint)		
Output switch = 1 (PNPN)		
F Clear AIC		

ACCUMULATOR CONTENTS:

P1	P2	P3	P4
-W	-W	-W	-W
N1	N2	N3	N4
-W	-W	-W	-W

W = WINDOW register data

(24) TAIC

The TAIC step executes the additions indicated in LOAR step and tests whether the rectangle just defined has any area in common with the current window. If there is area in common, the AIC flip-flop is set.

ROM BITS:	POKES:	PC:
ST Box edge F Allow AIC to set	Arbitrary	Jump WAIT

ACCUMULATOR CONTENTS:

Same as LOAR

3.3.4 Window Set Up

If load or box is not called for, the ENDN and ENDR state initiates dispatch 3. Dispatch 3 dispatches to one of four states depending on the mode established by the directive. If 3D curve is indicated, several special curve states are entered. The first one, called CURV, puts the old end of the line into the INSTANCE register for possible later use. If the old endpoint is negative, the data is complimented and corrected for twos compliment arithmetic by two ensuing steps. If the clipper is in curve mode, a check is made after clipping and division to see if the line drawn extends from the forward pyramid of vision into the rear pyramid of vision. If so, an additional clipping and output process will be required. CRET makes this check. If an additional pass is required, as indicated by the CUR flip-flop, CFEU initiates this pass by checking to see if the Z data for the new point is positive. The second pass then proceeds as the first.

If 3D not curve is indicated, the clipper goes directly into the WIN3 (3-dimensional window) state. If 2 D is indicated, the clipper goes into the WIN2 (2-dimensional window) state. In the contradictory case of 2D curve, a spare state is provided which returns the clipper to WAIT. From the windowing steps, the clipper goes to dispatch 5 and the clipping process.

(25) CURV

In this step the old end of the line is put into the INSTANCE register for possible later use (i.e., if another pass is required). A check is made to see if the old end of the line is negative, which would indicate that the clipper has been using complimented data.

ROM BITS:	POKES:	PC:
S Write Register address = 3 (INSTANCE) L Output switch = 3 (PPPP) F Allow CUR to set	Arbitrary	Jump COMP if test 202 (ZP-) otherwise WIN3

ACCUMULATOR CONTENTS:

P1	P2	P3	P4
P1	P2	P3	P4
N1	N2	N3	N4
N1	N2	N3	N4

P to INSTANCE

(32) COMP

If the CURV step discovers that PZ is negative, or the CFEU step discovers that NZ is negative, the COMP (compliment) state is entered. This step converts X,Y,Z to -X,-Y,-Z. Both ends of the line are complimented, so the pyramid of vision is effectively reversed.

ROM BITS:	POKES:	PC:
R Inject P odd (next pulse)		PC+1
Inject P even (next pulse)	Clear PU,NU	
Inject N odd (next pulse)		
Inject N even (next pulse)		
L Lower switch = 4 (select compliment)		

ACCUMULATOR CONTENTS:

0	0	0	0
-P1	-P2	-P3	-P4
0	0	0	0
-N1	-N2	-N3	-N4

(33) COMQ

COMQ performs the injections set up by COMP and transfers control to WIN3.

ROM BITS:	POKES:	PC:
L Sum switch = 3 (select full sum)	All	Jump WIN3

ACCUMULATOR CONTENTS:

-P1	-P2	-P3	-P4
-P1	-P2	-P3	-P4
-N1	-N2	-N3	-N4
-N1	-N2	-N3	-N4

Note: The following two steps pertain to the additional pass required if the CUR flip-flop is set.

(1) CFET

After each clipping and division process is completed, the clipper enters the CFET state to check if the CUR flip-flop has been set. If so, another pass is instigated; if not, the clipper returns to the WAIT state.

ROM BITS:	POKES:	PC:
S Register true output to AC Register address = 0 (SAVE) L Lower switch = 2 (select joint) Upper switch = 2 (select joint)	A11	Jump CFEU if YES = 2 NO = 0 test = 2 SEL = 2, otherwise PC+1

ACCUMULATOR CONTENTS:

X	X	X	X
X	X	X	X
S2	S1	S4	S3
S1	S2	S3	S4

(31) CFEU

CREU loads the P portion of the working registers with the data from the INSTANCE registers (which is stored there during the CURV state.). If ZN is negative, control goes to the COMP step, otherwise control is transferred directly to the WIN3 step

ROM BITS:	POKES:	PC:
S Register true output to AC Register address = 3 (INSTANCE) L Lower switch = 2 (select joint) Upper switch = 2 (select joint) F Clear CUR	ROM address 5 (poke P)	Jump WIN3 unless test 201 (ZN-)

ACCUMULATOR CONTENTS:

P2	P1	P4	P3
P1	P2	P3	P4
X	X	X	X
X	X	X	X

Note: P was loaded into the INSTANCE register by the CURV step.

(26) WIN3

Three-dimensional windowing requires that coordinates be converted from the X,Y,Z form in which they were given to the Clipping Divider into the form [X-(-Z),X-(+Z),Y-(-Z), Y-(+Z)]. However, in order to make both the line X = +Z and the line X = -Z be in the pyramid of vision, the actual conversion is: [X-(-Z),X-(+Z)-1,Y-(-Z,Y-(+Z)-1]. The two insertions of -1 are accomplished by not injecting the carry required by twos compliment arithmetic. WIN3 transposes the data and compliments one of the Z values to set up the accumulators for the forthcoming EDGE step.

ROM BITS:	POKES:	PC:
L Sum switch = 5 (select transpose) Lower switch = 3 (transpose and compliment)	Poke transpose and compliment	Dispatch 5 (SETP)

ACCUMULATOR CONTENTS:

P1	P1	P3	P3
P2	-P2	P4	-P4
N1	N1	N3	N3
N2	-N2	N4	-N4

1 = X, 2 = Z, 3 = Y, 4 = Z

(30) WIN2

The WIN2 step calls up the compliment of the data in the WINDOW register to form the values: [X-WL, X-WR, Y-WB, Y-WT]. This process sets up the data in the accumulators for the EDGE step.

ROM BITS:	POKES:	PC:
R Inject N odd (next pulse) Inject P odd (next pulse) S Register compliment output to AC Register address = 2 (WINDOW) L Lower switch = 2 (select joint)	Poke transpose and compliment	Dispatch 5 (SETP)

ACCUMULATOR CONTENTS:

P1	P2	P3	P4
-W1	-W2	-W3	-W4
N1	N2	N3	N4
-W1	-W2	-W3	-W4

W = WINDOW register data

(27) SPAI

This step is provided to detect the meaningless combination of 2D curve. It does nothing and returns control to the WAIT state.

3.3.5 Clipping

After the windowing steps are completed, control is transferred to dispatch 5. Dispatch 5 has two exits. One is used if a setpoint is indicated and the other if a line (not setpoint) is directed. In the not setpoint case, the steps involved in clipping and division are performed.

(34) SETP

In the SETP step, the new end of the line is checked against the window. If it is on, the Channel Control HIT bit is set. The SAVE register was set previous to windowing with information in regular coordinates. SETP does not change this information in the SAVE register.

	ROM BITS:	POKES:	PC:
F	Allow SETPOINT HIT Clear AGREE Clear CUR	Arbitrary	Dispatch 0 (CFET)

ACCUMULATOR CONTENTS

No Change

(35) EDGE

The EDGE step adds the accumulators together to form the sum indicated in the WIN2 and WIN3 steps. The carry injection that was set up in the windowing steps os also performed.

	ROM BITS:	POKES:	PC:
L	Sum Switch = 3 (select full sum)	All	PC+1
ST	Edge		

ACCUMULATOR CONTENTS

	XP-WL	XP-WR	YP-WB	YP-WT		XP+ZP	XP-ZP	YP+ZP	YP-ZP
	---	---	---	---		---	---	---	---
2D	XN-WL	XN-WR	YN-WB	YN-WT	3D	XN+ZN	XN-ZN	YN+ZN	YN-ZN
	---	---	---	---		---	---	---	---

Selection to lowers controlled by Poke Control as follows:

Both on	N on	P on	Both off
PL PR PB PT	PL PR PB PT	PL PR PB PT	PL PR PB PT
PL PR PB PT	NL NR NB NT	PL PR PB PT	NL NR NB NT
NL NR NB NT	NL NR NB NT	NL NR NB NT	NL NR NB NT
NL NR NB NT	NL NR NB NT	PL PR PB PT	PL PR PB PT

PL = XP-WLorXP+ZP;PR = XP-WRorXP-ZP; etc.

(36) CLIP

The CLIP step is a repeated step. The values of the previous and new points for both X and Y in window edge coordinates are added and the half sum taken. The half sum replaces either the upper or lower accumulators in such a way as to home in on the point at which the line intersects the window. At the end of the clipping process, the point of intersection of the line and edge of the window will have been found. Thus, the two ends of the visible segment of the line will be in the accumulators at the end of the CLIP step.

At any iteration of the CLIP step, a non-trivial rejection may be detected, in which case, the clipper returns to the WAIT state.

Before clipping, the accumulator registers were loaded as indicated in the EDGE step. Thus, if either the previous or new point was on the window, the clipping process leaves the data unchanged. The P accumulators clip the previous point by loading P into the upper registers and N into the lower registers. The N accumulators have N loaded into the uppers and P into the lowers which allows them to clip the N point.

ROM BITS:	POKES:	PC:
R Outround odd Outround even E Enable odd - Enable even +	All	Dispatch 0 (CFET) if clip reject otherwise stay in CLIP until DONE then PC+1

ACCUMULATOR CONTENTS:

1/2 upper + lower placed in the P registers as follows:

Upper: 1. Thru clipping P
2. P off on the same side as half point

Lower: 1. Half point on
2. Half point off on the same side as N

The situation for the N registers is obtained by substituting N for P and P for N.

3.3.6 Page Coordinates to Memory

After the CLIP step, the clipper goes into an output sequence that may result in writing into memory certain data in page coordinates before it is converted into scope coordinates. The TANC step which immediately follows clipping is the first step of this sequence. If the clipper is in 3D mode, or if no page coordinate output is required, control is transferred from TANC to dispatch 6 which dispatches to one of several steps. If no output is required, the clipper will go from dispatch 6 to VIEW which calls for the content of the VIEWPORT register. VIEW branches into either an address to memory (ADTM) step or directly to the division step. If output is required, TANC goes into several steps which are performed depending on the mode and nature of the data. These steps all eventually lead to the VIEW step and on to division.

During the entire page coordinate output sequence, the A side accumulators are not affected and remain in window edge coordinates. Only the B side accumulators contents are put in page coordinates. It should also be noted that output to the external devices comes only from the A side output lines. These lines are capable, however, of selecting data from the B side accumulators.

(37) TANC

The sequence starts with the TANC step which, with COUT and COUU, converts data from window edge coordinates to regular coordinates. The TANC step places the compliment value of the clipped P coordinate for the right side in the lower register and the coordinate for the left side in the corresponding upper register.

ROM BITS:	POKES:	PC:
R Inject P odd (next pulse) Inject N odd (next pulse) L Sum switch = 5 (select transpose) Lower switch = 3 (transpose and compliment)	ROM address 1 (transpose and complement)	Dispatch 6 if test 32 (3D or no output)

ACCUMULATOR CONTENTS:

PCL	PCL	PCB	PCB
PCR	-PCR	PCT	-PCT
NCL	NCL	NCB	PCB
NCR	-NCR	NCT	-NCT

PCL = Previous point clipped with respect to the left window edge.

(40) COUT

The COUT step loads the lower B registers with data from the WINDOW register so that a conversion can be made to page coordinates.

ROM BITS:	POKES:	PC:
R Inject P even (next pulse) Inject N even (next pulse) S Register true output to AC Register address = 2 (WINDOW) L Sum switch = 1 (select lower) Lower switch = 2 (select joint)	ROM address 4 (poke evens)	PC+1

ACCUMULATOR CONTENTS:

X	PCR	X	PCT
X	WR	X	WT
X	NCR	X	NCT
X	WR	X	WT

(41) COUU

The COUU step performs the additions indicated in the COUT step.

ROM BITS:	POKES:	PC:
L Sum switch = 3 (select full sum)	ROM address 4 (evens)	Jump NPTM if test 21 (dot) otherwise PPTM

ACCUMULATOR CONTENTS:

X	PX	X	PY
X	PX	X	PY
X	NX	X	NY
X	NX	X	NY

(42) PPTM

PPTM outputs the previous point to memory and raises the memory output flag.

ROM BITS:	POKES:	PC:
R Inject N odd (next pulse) L Sum switch = 4 (even to both uppers) Output switch = 1 (PNPN) I Raise outflag for memory data Output wait	ROM address 4 (evens)	Jump NXTM if test 30 (3D) otherwise PC+1

ACCUMULATOR CONTENTS:

No change. PUBX, PUBY to memory

(44) ZTOS

ZTOS gives depth cueing data to the scope. This means that the three-dimensional Z value must be sent to the intensity register of the scope. If this is called for, the ZTOS step must be performed.

ROM BITS:	POKES:	PC:
R Inject P odd (next pulse) Inject N odd (next pulse) L output switch = 5 (PPNN) I Adage take Z only Raise outflag for scope Output wait	Poke Z and READ Poke ROM address 4 (evens)	Jump VIEW

ACCUMULATOR CONTENTS:

No change. PBZ, NBZ to memory

(45) PXTM

The PXTM step is executed if the clipper is in 3D mode. It writes the previous X and Y values in memory and sets up the accumulator so that the PPTM step to which the PXTM step jumps will write the Z values in memory.

ROM BITS:	POKES:	PC:
L Output switch = 1 (PNPN) I Raise outflag for memory data Output wait	ROM address 4 (evens)	Jump PPTM

ACCUMULATOR CONTENTS:

X	PZ	X	PZ
X	PZ	Z	PZ
X	NZ	X	NZ
X	NZ	X	NZ

(46) NXTM

NXTM performs the same task as PXTM for the new point.

ROM BITS:	POKES:	PC:
L Output switch = (NNNN) I Raise outflag for memory data Output wait	ROM address 4 (evens)	Jump NPTM

ACCUMULATOR CONTENTS:

No change.

(47) VIEW

This step calls up the content of the VIEWPORT register and loads the data in the B side accumulators to set up the Clipping Divider up for the division step. It should be remembered that the data in the A side of the accumulators are left in window coordinates.

ROM BITS:	POKES:	PC:
S Register true output to AC Register address = 1 (VIEWPORT) L Upper switch = 2 (select joint) Lower switch = 2 (select joint) F Force SETPOINT HIT	Poke ROM address 4 (evens)	Jump DIVI unless test 104 (address to memory) then ADTM

ACCUMULATOR CONTENTS:

PCL	VL	PCB	VB
PCR	VR	PCT	VT
NCL	VL	NCB	VB
NCR	VR	NCT	VT

(50) ADTM

The step ADTM simply transfers the contents of the clipper NAME register to memory. It does this with using the accumulators.

ROM BITS:	POKES:	PC:
S Register true output to AC Register address = 4(NAME) L Output switch = 4 (RNRN) I Raise outflag for memory data Output wait	No	PC+1

3.3.7 Division

The division process consists of two steps designed to scale the clipped line and, if called for, prepare the scaled line for output. If no output is required, the division step will return control to WAIT. If output is required, the clipper goes to dispatch 7 which controls the output sequence.

(51) DIVI

The DIVI step is iterated until the scaled output value is found. The divide step terminates when all four divisions have been completed. A division is complete when 1. the two answer accumulators (B side) contain the same value or 2. the window edge accumulators form an all propagate sum. These conditions are detected by the logic found on the Done card. When DIVI is complete, the B side accumulators contain scope coordinate answers, and the A side accumulators contain zero.

<u>ROM BITS</u>:	<u>POKES</u>:	<u>PC</u>:
E Enable odd +	All	Jump when DONE
Enable odd -		to dispatch 0
ST Divide - type shift allowed		if no output,
Divide		otherwise DDIS

<u>ACCUMULATOR CONTENTS</u>:

1/2 uppers + lowers is placed in uppers if sum positive
lowers if sum negative

(52) DDIS

This step checks the state of the flip-flop AGREE which indicates whether a setpoint instruction need be sent to the scope. It then initiates and conditions dispatch 7 and clears the AGREE flip-flop.

ROM BITS:	POKES:	PC:
F Clear AGREE	Arbitrary	Dispatch 7 (PSTM)

ACCUMULATOR CONTENTS:

0	PSX	0	PSY
0	PSX	0	PSY
0	NSX	0	NSY
0	NSX	0	NSY

PS = previous point, scope coordinates.

3.3.8 Output

Dispatch 7, which is conditioned by the DDIS step, controls the output sequence. Output can go to memory, to the scope, or both. Since the previous pass through the algorithm, the flip-flop AGREE remembers whether the scope coordinate setpoint, at which the display is resting, matches the page coordinate setpoint in the clipper. If so, output to the scope for a new line need only draw the line. If not, both scope set point and line drawing commands must be given the scope. The scope setpoint is issued by PSTS and the line drawn by NSTS. NSTS sets the AGREE flip-flop if the new end of the line is on the screen. AGREE is cleared by the DDIS after the decision as to dispatch has been made. AGREE is also cleared by any operation which changes the WINDOW or VIEWPORT registers or during the setpoint operation of the clipper.

(53) PSTM

This step writes the coordinates of the previous point in memory. It is omitted if a dot is to be drawn.

ROM BITS:	POKES:	PC:
L Sum switch = 4 (even to both uppers) Output switch = 1 (PNPN) I Raise Outflag for memory data Output wait	Arbitrary	PC+1

ACCUMULATOR CONTENTS:

No change. PUBX, PUBY to memory.

(54) NSTM

This step writes the coordinates of the new point in memory.

ROM BITS:	POKES:	PC:
L Sum switch = 4 (even to both uppers) Output switch = 2 (NNNN) I Raise outflag for memory data Output wait	Arbitrary	Dispatch 0 (DFET) unless test 41 (scaled to scope)

ACCUMULATOR CONTENTS:

No change. NUBX, NUBY to memory

(55) PSTS

The PSTS step is executed if scaled output to scope is desired and the AGREE flip-flop has been cleared previous to the DDIS step. This step issues a setpoint command to the scope.

ROM BITS:	POKES:	PC:
L Sum switch = 4 (evens to both uppers) Output switch = 1 (PNPN) I DOVG Adage take X and Y (and Z if no depth) Output wait Raise outflag for scope	Arbitrary	PC+1

ACCUMULATOR CONTENTS:

No change. PUBX, PUBY to scope

(56) NSTS

The NSTS step sends the coordinates of the new point to the scope which results in the drawing of the line.

ROM BITS:	POKES:	PC:
L Sum switch = 4 (evens to both uppers) Output switch = 2 (NNNN) F Allow AGREE to set I DOVG Adage take X,Y, and Z Raise outflag for scope Output wait	Arbitrary	Dispatch 0 (CFET)

3.3.9 Boxing

The boxing process allows the display of subpictures. The INSTANCE register is loaded with the page coordinates of the subpicture to be displayed. These coordinates may be thought of as defining a subwindow (instance). The instance is then compared with the existing window. The resulting data is manipulated according to the case of intersection and prepared for the division step. Boxing division is similar to the normal clipper division process. If an edge of the subwindow is outside the corresponding edge of the existing window, a new subwindow must be defined. If the subwindow edge is inside the existing window edge, a new viewport must be defined so that the subpicture will be displayed on the correct portion of the scope.

The boxing sequence (shown on the right side of sheet 1 of the algorithm flow diagram (101123-950)) first loads the coordinates of the area to be mapped into the INSTANCE register and puts the data in window edge coordinates. The steps following prepare the data for division. The steps after the division step test cases of intersection and write new windows and viewports as required.

The format used in detailing the algorithm will change somewhat in the following discussion. There are 8 cases of intersection in X and 8 in Y which results in 64 possible intersections. The accumulator contents for many of the steps depend on the case of intersection. The situation, however, is parallel for X and Y. For this reason, the accumulator contents will be delineated by case in a separate chart.

(17) BOXR

At the end of the LOAD step, some of the accumulators are loaded with data from the INSTANCE register. The BOXR step loads the new data in the accumulators into the INSTANCE register. BOXR and LOAD thus serve to exchange the contents of the accumulator and INSTANCE register.

ROM BITS:	POKES:	PC:
S Write Register address = 0 (INSTANCE) L Sum switch = 6 (select new data) Output switch = 1 (PNPN)	No	Jump BWIN

ACCUMULATOR CONTENTS:

P2	I1	P4	I3
I1	P1	I1	P1
I2	N1	I4	N3
N2	I2	N4	I4

(57) BWIN

The BWIN step is almost exactly like the WIN2 step. The contents of the WINDOW register, in compliment form, are brought up and loaded into the lower registers of the accumulators.

ROM BITS:	POKES:	PC:
R Inject P odd (next pulse) Inject N odd (next pulse) S Register compliment output to AC Register address = 2 (WINDOW) L Sum switch = 7 (select old data) Lower switch = 2 (select joint)	All	PC+1

ACCUMULATOR CONTENTS:

I1	I1	I3	I3
-W1	-WR	-WB	-WT
I2	I2	I4	I4
-WL	-WR	-WB	-WT

(60) BDEG

This is the edging step for the boxing process. The instance is added to the compliment of the window, thus putting the instance in window edge coordinates.

ROM BITS:	POKES:	PC:

L Sum switch = 3 (select full sum)
ST Edge
Boxedge
	All	Jump WAIT if
		YES = 1
		SEL = 2
		test = 100 (trivial rejection) otherwise PC+1

(61) BTRN

In the case that an edge of the instance is outside the window, the data for that edge are transposed.

ROM BITS:	POKES:	PC:
L Sum switch = 7 (select old data)	ONLY OFF ROM address 7 (box transpose)	PC+1

(62) BTRU

If the instance edge is reversed in respect to the window edge (i.e., the left edge is outside the right edge of the window or the right edge outside the left edge of the window), the data in the lower registers of the accumulator handling that coordinate of data are loaded into the upper register on the opposite side (odds go to evens and evens to odds).

ROM BITS:	POKES:	PC:
L Sum switch = 1 (select lowers)	ONLY REV ROM address 2 (uppers)	PC+1

(63) BSWU

The BSWU step swaps the uppers in those cases where they have just been loaded by the last step.

ROM BITS:	POKES:	PC:
L Sum switch = 2 (select uppers)	ONLY REV ROM address 2 (uppers)	PC+1

(64) BMAS

BMAS loads the new value of the INSTANCE register (put in by the LOAD and BOXR steps) into the even registers of the accumulator. The data in the upper portions of the even accumulators is simultaneously loaded into the lower registers of the odd portion.

ROM BITS:	POKES:	PC:
S Register true output to AC Register address = 3 (INSTANCE) L Upper switch = 2 (select joint) Sum switch = 2 (select uppers) Lower switch = 5 (transpose and input)	ROM address 1 (transpose and compliment)	PC+1

(65) BVPT

In the cases where a new viewport must be written (i.e., when an edge of the instance is within the window), the value of the VIEWPORT register is called up and loaded into the appropriate quadrant.

ROM BITS:	POKES:	PC:
S Register true output to AC Register address = 1 (VIEWPORT) L Upper switch = 2 (select joint) Sum switch = 2 (select upper) Lower switch = 5 (transpose and input)	ONLY ON ROM address 4 (evens)	PC+1

(66) BDIV

The BDIV step does the division as in the DIVI step and produces the scope coordinates of the new viewport or the page coordinates of the new window for each edge of the instance.

ROM BITS:	POKES:	PC:
E Enable odd+ Enable odd- ST Divide - type shift allowed Divide	All	Repeat until DONE(SEL=2) then PC+1

(67) BOLM

After the division step, certain checks must be made to check the cases of intersection. As a preliminary to these steps, the old master from the INSTANCE register is loaded into the upper registers and the answers from BDIV are loaded into the lowers.

ROM BITS:	POKES:	PC:
S Register true output to AC Register address=3(INSTANCE) L Upper switch=2(select joint) Sum switch=4(even to both uppers)	All	PC+1

(70) BRPM

BRPM replaces the master with the new window edge if the edge is off.

ROM BITS:	POKES:	PC:
L Sum switch = 1 (select lower)	ONLY OFF ROM address 2 (uppers)	PC+1

(71) BWNW

This step writes the new window in the WINDOW register.

ROM BITS:	POKES:	PC:
S Write Register address=2(WINDOW) L Sum switch=2 (select upper) Output switch=1 (select PNPN)	ROM address 2 (uppers)	PC+1

(72) BOLV

BOLV loads the value in the VIEWPORT register into the upper registers as a preliminary to checks for special cases and loading of the new data into the VIEWPORT register.

ROM BITS:	POKES:	PC:
S Register true output to AC Register address=1(VIEWPORT) L Upper switch = 2 (Select joint)	ROM address 2 (uppers)	PC+1

(73) BSWV

BSWV swaps the upper registers if the instance is off backwards (reversed).

ROM BITS:	POKES:	PC:
L Lower switch=2(select joint)	ONLY REV ROM address 2 (uppers)	PC+1

(74) BRPV

The old viewport values are replaced with new values in the cases where the instance is on the window.

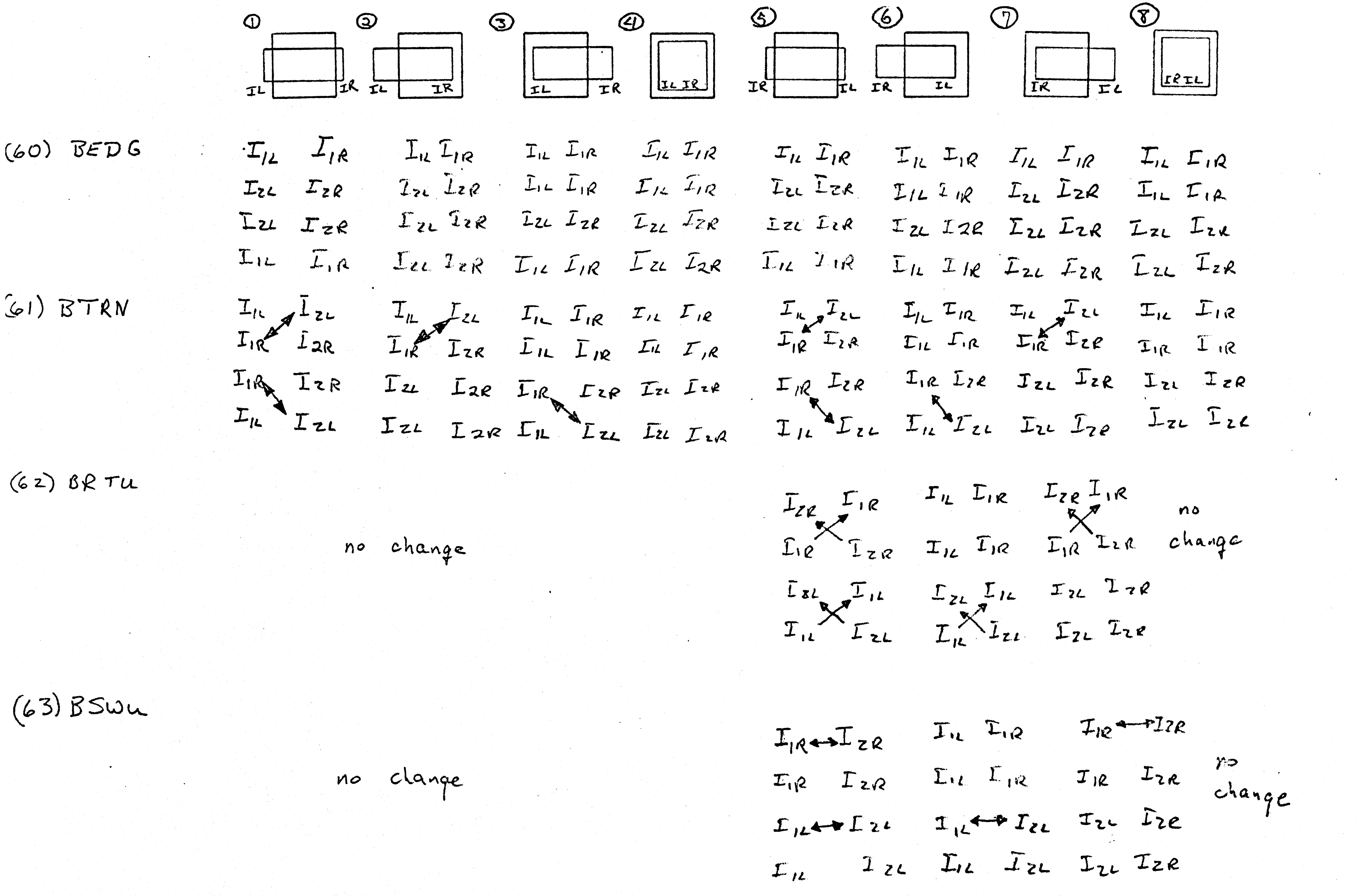
ROM BITS:	POKES:	PC:
L Lower switch=1 (select other sum)	ONLY REV ROM address=2 (uppers)	PC+1

(75) BWNV

BWNV loads the VIEWPORT register with the new VIEWPORT and concludes the boxing sequence.

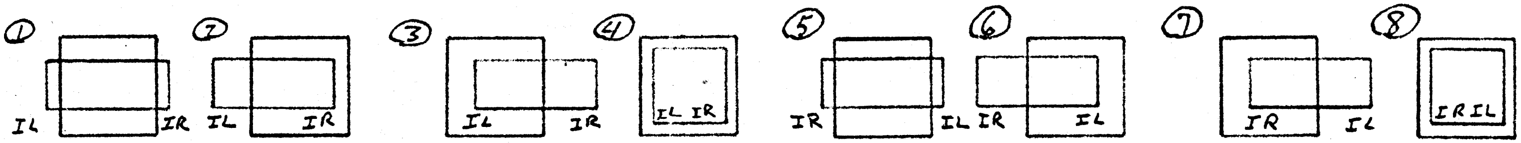
ROM BITS:	POKES:	PC:
S Write Register address=1 (VIEWPORT) L Sum switch=2(select upper) Output switch=1(PNPN)	Arbitrary	Jump WAIT

Note: Accumulator contents for boxing steps will be shown on charts delineating the cases.



1 2 3 4 5 6 7 8
IL IR
IL IR
IL IR
IL IR
IR IL
IR IL
IR IL
IR IL
(60) BEDG
I1L I1R I1L I1R I1L I1R I1L I1R I1L I1R I1L I1R I1L I1R I1L I1R
I2L I2R I2L I2R I1L I1R I1L I1R I2L I2R I1L I1R I2L I2R I1L I1R
I2L I2R I2L I2R I2L I2R I2L I2R I2L I2R I2L I2R I2L I2R I2L I2R
I1L I1R I2L I2R I1L I1R I2L I2R I1L I1R I1L I1R I2L I2R I2L I2R
(61) BTRN
I1L I2L I1L I2L I1L I1R I1L I1R I1L I2L I1L I1R I1L I2L I1L I1R
I1R I2R I1R I2R I1L I1R I1L I1R I1R I2R I1L I1R I1R I2R I1R I1R
I1R I2R I2L I2R I1R I2R I2L I2R I1R I2R I1R I2R I2L I2R I2L I2R
I1L I2L I2L I2R I1L I2L I2L I2R I1L I2L I1L I2L I2L I2R I2L I2R
(62) BRTU
no change
I2R I1R I1L I1R I2R I1R no change
I1R I2R I1L I1R I1R I2R
I2L I1L I2L I1L I2L I2R
I1L I2L I1R I2L I2L I2R
(63) BSWU
no change
I1R I2R I1L I1R I1R I2R no change
I1R I2R I1L I1R I1R I2R
I1L I2L I1L I2L I2L I2R
I1L I2L I1L I2L I2L I2R

Figure 3.1 Accumulator Contents for Boxing



(64) BMAS

(m from INSTANCE register)

I_{1L} m_1	I_{1L} m_1	I_{1L} m_1	I_{1L} m_1	I_{1R} m_1	I_{1L} m_1	I_{1R} m_1	I_{1L} m_1
I_{2L} m_2	I_{2L} m_2	I_{1R} m_2	I_{1R} m_2	I_{2R} m_2	I_{1R} m_2	I_{2R} m_2	I_{1R} m_2
I_{1R} m_1	I_{2L} m_1	I_{1R} m_1	I_{2L} m_1	I_{1L} m_1	I_{1L} m_1	I_{2L} m_1	I_{2L} m_1
I_{2R} m_2	I_{2R} m_2	I_{2R} m_2	I_{2R} m_2	I_{2L} m_2	I_{2L} m_2	I_{2R} m_2	I_{2R} m_1

(65) BVPT

(V from VIEWPORT register)

I_{1L} m_1	I_{1L} m_1	I_{1L} V_1	I_{1L} V_1	I_{1R} m_1	I_{1L} V_1	I_{1R} m_1	I_{1L} V_1
I_{2L} m_2	I_{2L} m_2	I_{1R} V_2	I_{1R} V_2	I_{2R} m_2	I_{1R} V_2	I_{2R} m_2	I_{1R} V_2
I_{1R} m_1	I_{2L} V_1	I_{1R} m_1	I_{2L} V_1	I_{1L} m_1	I_{1L} m_1	I_{2L} V_1	I_{2L} V_1
I_{2R} m_2	I_{2R} V_2	I_{2R} m_2	I_{2R} V_2	I_{2L} m_2	I_{2L} m_2	I_{2R} V_2	I_{2R} V_2

For the next steps only case 2 will be concidered (case 7 will be mentioned where ONLY REV is different)

(66) BDIV

after division

- NM_1
- NM_1
- NV_2
- NV_2

(67) BOLM

m from INSTANCE

m_2 m_1

NM_1 NM_1

m_2 m_1

NV_2 NV_2

(70) BRPM

NM_1 NM_1

NM_1 NM_1

m_2 m_1

NV_2 NV_2

(71) BWNW

NM_1 NM_1

NM_1 NM_1

m_1 m_2

NV_2 NV_2

NM_1 , m_2 to new WINDOW

(72) BOLV

V from VIEWPORT

V_2 V_1

NM_1 NM_2

V_2 V_1

NV_2 NV_2

Figure 3.1 Accummulator Contents for Boxing

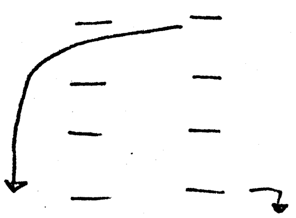
(73) BSWV

case 2	case 7
V_2 V_1	$U_1 \leftrightarrow V_2$
NM_1 NM_1	NM_1 NM_1
V_2 V_1	V_2 V_1
NU_2 NV_2	NV_2 NV_2

(74) BRPV

case 2	case 7
V_2 V_1	V_1 V_2
NM_1 NM_1	NM_1 NM_1
NV_2 NU_2	NU_2 NV_2
NU_2 NU_2	NU_2 NU_2

(75) BWNV



case 2: U_1 and NU_2 to VIEWPORT

case 7: U_2 and NU_2 to VIEWPORT

Figure 31 Accummulator Contents for Boxing

CLIPPER BACK PANEL

4.1 Introduction

The cards containing the logical circuitry of the Clipping Divider are contained in three card cages, each capable of holding 24 cards. The top cage contains cards relating to the data paths for the X coordinates; the bottom cage contains corresponding cards for the Y coordinates. The center cage contains the cards which provide the control logic needed by the Clipping Divider.

4.2 Reference

E & S Block Diagram No. 101123-300, Clipper Back Panel

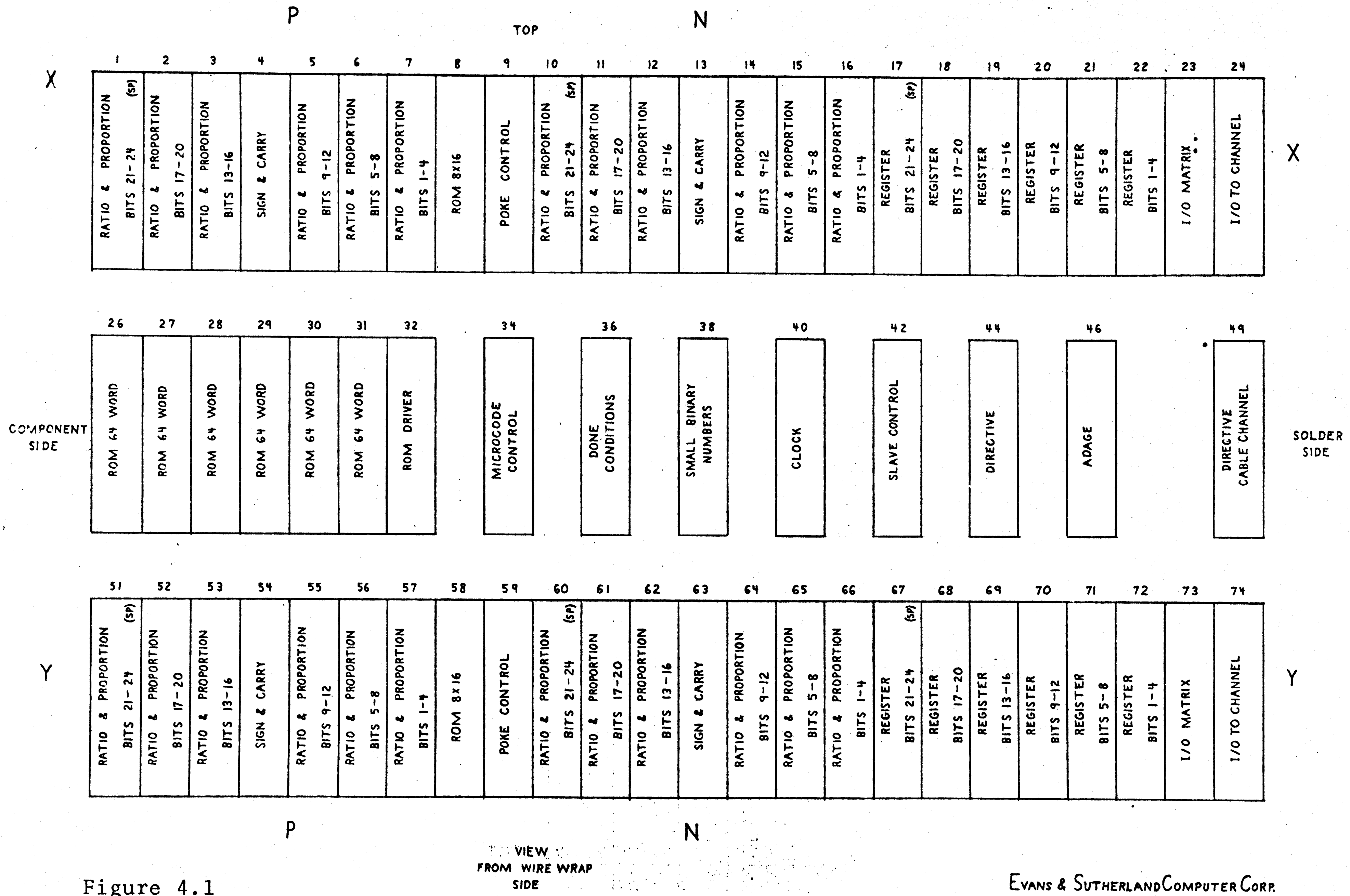
Clipper Back Panel Wirelist

4.3 Card Placement and Intercard Connections

In the following discussion we will first examine the placement of the individual cards in the cages and, on a very general level, the function of each of the cards. Details will then be given on the intercard connections as they relate to the functions of the Clipping Divider.

4.3.1 Card Placement

Figure 4.1 shows the stuffing chart for the various cards



1	2	3	4	5	6	7	8	9	10	11	12
RATIO & PROPORTION BITS 21-24 (SP)	RATIO & PROPORTION BITS 17-20	RATIO & PROPORTION BITS 13-16	SIGN & CARRY	RATIO & PROPORTION BITS 9-12	RATIO & PROPORTION BITS 5-8	RATIO & PROPORTION BITS 1-4	ROM 8X16	POKE CONTROL	RATIO & PROPORTION BITS 21-24 (SP)	RATIO & PROPORTION BITS 17-20	RATIO & PROPORTION BITS 13-16

13	14	15	16	17	18	19	20	21	22	23	24
SIGN & CARRY	RATIO & PROPORTION BITS 9-12	RATIO & PROPORTION BITS 5-8	RATIO & PROPORTION BITS 1-4	REGISTER BITS 21-24 (SP)	REGISTER BITS 17-20	REGISTER BITS 13-16	REGISTER BITS 9-12	REGISTER BITS 5-8	REGISTER BITS 1-4	I/O MATRIX	I/O TO CHANNEL

26	27	28	29	30	31	32	34	36	38	40	42	44	46	49
ROM 64 WORD	ROM 64 WORD	ROM 64 WORD	ROM 64 WORD	ROM 64 WORD	ROM 64 WORD	ROM DRIVER	MICROCODE CONTROL	DONE CONDITIONS	SMALL BINARY NUMBERS	CLOCK	SLAVE CONTROL	DIRECTIVE	ADAGE	DIRECTIVE CABLE CHANNEL

51	52	53	54	55	56	57	58	59	60	61	62
RATIO & PROPORTION BITS 21-24 (SP)	RATIO & PROPORTION BITS 17-20	RATIO & PROPORTION BITS 13-16	SIGN & CARRY	RATIO & PROPORTION BITS 9-12	RATIO & PROPORTION BITS 5-8	RATIO & PROPORTION BITS 1-4	ROM 8X16	POKE CONTROL	RATIO & PROPORTION BITS 21-24 (SP)	RATIO & PROPORTION BITS 17-20	RATIO & PROPORTION BITS 13-16

63	64	65	66	67	68	69	70	71	72	73	74
SIGN & CARRY	RATIO & PROPORTION BITS 9-12	RATIO & PROPORTION BITS 5-8	RATIO & PROPORTION BITS 1-4	REGISTER BITS 21-24 (SP)	REGISTER BITS 17-20	REGISTER BITS 13-16	REGISTER BITS 9-12	REGISTER BITS 5-8	REGISTER BITS 1-4	I/O MATRIX	I/O TO CHANNEL

Figure 4.1

EVANS & SUTHERLAND COMPUTER CORP.
CLIPPER LAYOUT

SHEET 1 OF 12

[illegible]

in the cages.

The upper and lower cages contain the accumulators, the storage registers and the associated poke and selection control cards. Each cage contains two groups of accumulator cards. One pertains to the P (previous) point and the other to the N (new) point. The cards contained in these cages are listed below.

Card Name and Number	Quantity	Function
Ratio & Proportion #101105	6/section	Constitute the basic arithmetic unit of the Clipper.
Sign & Carry #101106	1/section	Generates sign and carry information for the R & P cards.
Register #101107	6/cage	Contain the basic storage registers.
8 x 16 Read Only Memory #101109	1/cage	Controls selection switches for input and output of accumulator and storage registers.
Poke Control #101108	1/cage	Controls poking of the accumulator adders.
I/O Cables	2/cage	Connect the Clipper with external devices.

For systems of less than 24 bits, the low order bits are dropped in groups of 4. The data is then right justified and extra bits become a sign extension. For example, an 18 bit system will have one less R & P and one less Register card per section which results in a 20 bit field. Data are right justified and the two high order bits become a sign extension.

The center cage contains cards that pertain to control of data manipulation and the progression of the Clipper through the algorithm. These cards are listed below.

Card Name & Number	Quantity	Function
64 Word Read Only Memory #101110	6	A read only memory for selection of control signals for each step of the algorithm.
ROM Driver #101111	1	Selects and drives a word of Read Only Memory.
Microcode Control #101112	1	Contains program counter and test circuitry to determine the progression of the algorithm.
Small Binary Numbers #101126	1	An interface between the Directive and the Microcode Control cards used to control micro-instruction and dispatches.
Directive #101116	1	Registers and decodes the directive coming in on the directive input cable.
Done Conditions #101113	1	Determines when clipping or division has terminated.
Slave Control #101118	1	Holds selection information for up to 8 scopes and Z intensity information.
Adage Control #101132	1	Interfaces between the Clipper and the Adage CRT.
Clock #101117	1	Provides a clock impulse and communication between the Clipper and external devices.

4.3.2 I/O Data Paths

All input/output data for the Clipper is originated or terminated at the Register card. Cable 1 brings data into the Clipper from the Channel Control. Connector pins from the I/O cable are wired to the IN input bus of the Register cards. The OUTA output bus of the Register cards is in turn wired to the I/O cable. Inputs and outputs alternate in clusters of four down the cable.

Cable 2 brings data from the Matrix Multiplier. Connector pins of the cable are wired to the INA and INB input busses of the Register card alternating groups of four. Cable 2 is exclusively an input cable. If no Matrix Multiplier is included in the system, cable 2 contains data from the opposite coordinate and INA and INB are jumped together on the cable. In such a configuration, the cables in positions 24 and 73 contain X coordinate data and the cables in positions 74 and 23 contain Y coordinate data. This allows X and Y to be interchanged as indicated by the SWAP function of the Clipper.

4.3.3 Data Paths Between the Register and the R & P Cards

2-way selection switches choose between the IN or INA, INB input busses on the Register card. The selected input forms the DA,DB output bus. The DA bus is wired to the DB input of the R & P cards. This arrangement allows input data to be loaded into the upper registers of the R & P cards. Data is, however, interchanged (i.e. A data is loaded into the B side and B data is loaded into the A side.)

The RA output of the Register contains either the true or compliment form of the data that had been stored in a selected storage register. The RA output is wired to the RA input of both the N and P series of the R & P cards. Similarly, RB is wired to the RB input of the R & P cards. This allows data to be placed in the lower registers or interchanged and placed in the upper registers of the accumulators.

Figure 4.2 shows the data path connections between the Register and Ratio and Proportion cards.

Data paths between Register and Ratio and Proportion cards.

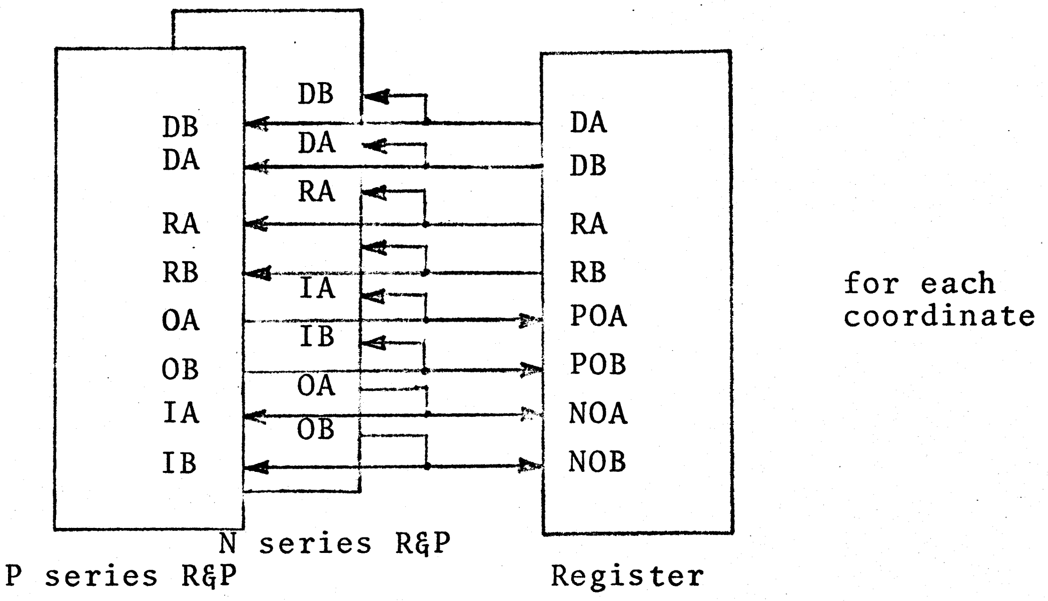


Figure 4.2

Output of the R & P cards is wired to the opposite set of R & P cards and to the Register card. The OA output of the N series R & P cards should be wired to the IA input of the P series cards and then to the NOA inputs of the Register card. The OA output of the P series is similarly wired to the IA of the N series and to the POA input of the Register cards. This allows data to be exchanged between the P and N sides of the accumulator and also allows the output of the R & P cards to be loaded into a selected storage register or placed on the output bus.

4.3.4 Sign and Carry Projection Paths

A special Sign and Carry card is used in connection with each set of R & P cards to project the sign of the sum to be computed and also to provide carry information to the adders of the accumulator. The control paths between the Sign and Carry and the Ratio and Proportion cards are shown in Figure 4.3 below.

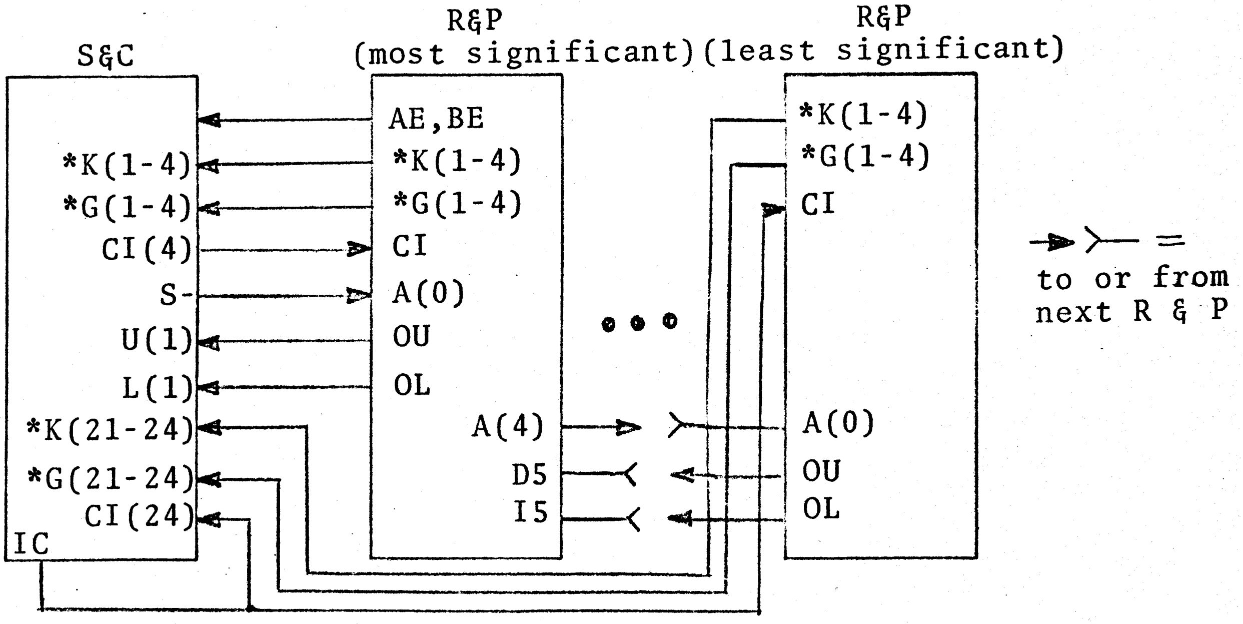


Figure 4.3

The carry generate (*AG,*BG) and carry kill (*AK,*BK) outputs of the R & P cards are wired to respective points on the S & C card. Outputs from a particular R & P card are wired to the S & C card inputs representing the group of digits handled by that R & P card.

The carry in signals developed by the S & C card are in turn wired to the respective carry in inputs for the adders of the R & P. For example, CIA8 from the S & C card is wired to CIA on the second most significant R & P card. The carry injection signals (ICA,ICB) used in fixing twos compliment arithmetic are wired to the low order R & P card carry inputs and back into the projection chain.

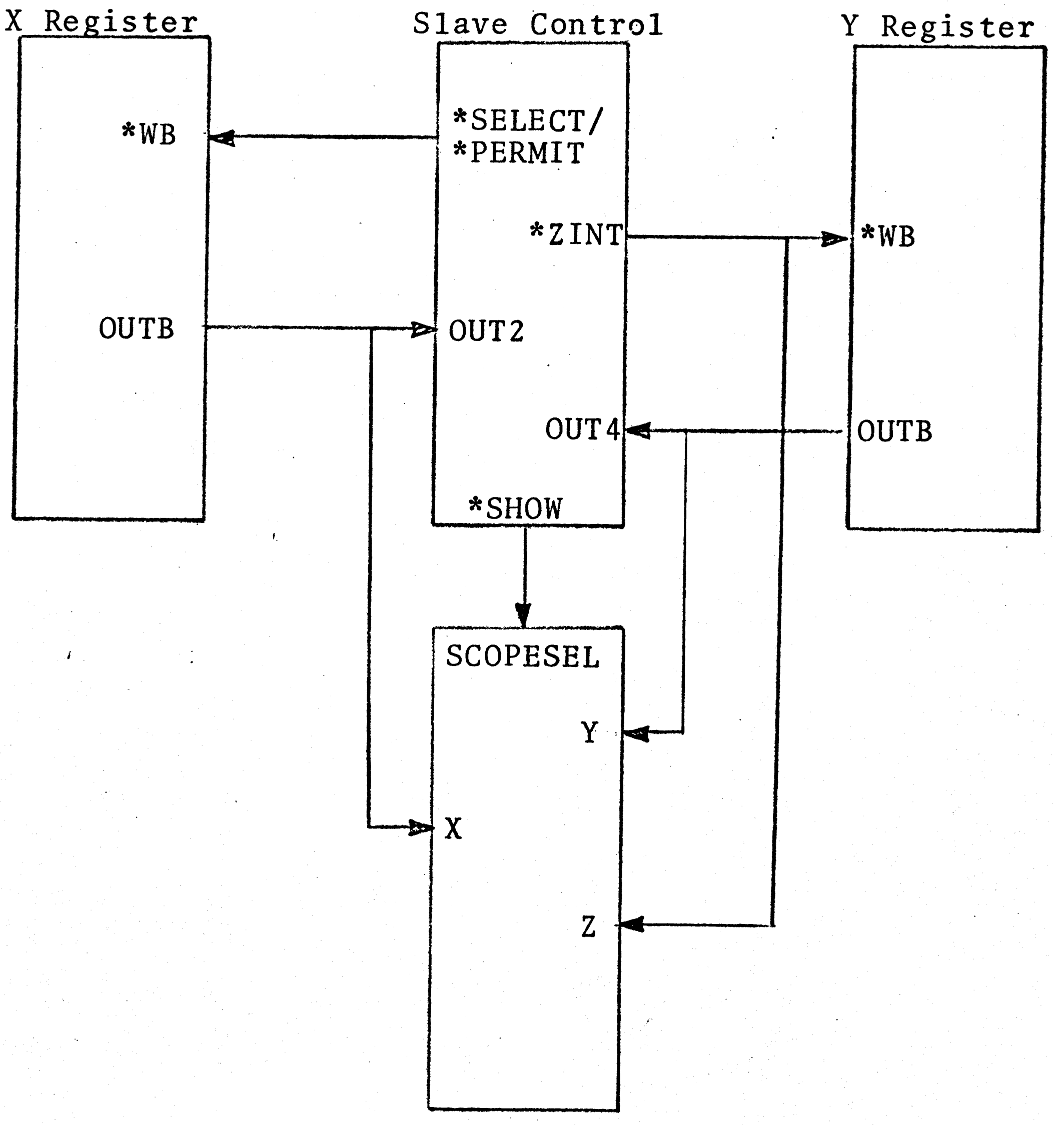
The high order output bits (OLA(1), OLB(1), etc.) of each of the registers on the R & P card handling bits 1-4 are wired to the sign input bits of the S & C card (LB(1) etc.). The predicted sign of the sum from the Sign and Carry card (SA- and SB-) are wired into the right shift paths of the R & P card (AAO,ABO). The right shift path is continued as the high order adder output bits (AA4,AB4) are wired to the low order output bits (AAO,ABO) on the next less significant card. The high order bit of the data of each register (OULA, OULB, etc.) is wired into the shift in input of the next most significant card.

4.3.5 Data Paths From the Storage Registers to the Adage Scopes

The right side (B side) output bus (OUTB) from the Register cards are wired to the Slave Control card for the purposes of storing the Z intensity and scope selection information to be passed to the scope. This information is sent to the Adage Interface card.

The left side output bus (OUTA) of the Register cards is wired, among other places, to the X and Y signals of the Adage Interface card. Main data paths are shown in Figure 4.4 below.

Data Output Paths



Adage Interface

Figure 4.4

The OUTB output bus (right side) of the Register is wired to the OUT2 input bus on the Slave Control card and the OUTB output of the Y Registers is wired to the OUT4 input bus on the Slave Control. The Z intensity (*ZINT) and selection (*SELECT) outputs of the Slave Control card are in turn wired to the *WB input bus to the Register card. This permits the data in the selection and intensity register on the Slave Control card to be loaded into the accumulators for later dispatch to memory. The *ZINT signals from the Slave Control are also wired to the Z inputs of the Adage Interface card to be delivered as a Z modulation data to the Adage unit. The selection outputs *SHOW are wired to the Adage Interface card signals (SCOPESEL) for the selection of slave scopes. It should be noted that the five low order bits of Z are not used in the Adage Interface card and that the high order bit is true while the others are inverted.

The left side of the register output bus (OUTA) is wired, in addition to other places, to the X and Y signals of the Adage Interface card. Only the upper 12 bits of the X,Y data (bits 19-20) are sent to the Adage Interface card. The low 3 bits of the X and Y data represented on the Adage Interface card, bits 12-14, are grounded on the back panel.

4.3.6 Register Card Counters

The counters on the Register cards are controlled by the logic located on the Done card. Four 4-bit ANGLE counters are located on the Y Register cards that occupy the middle four positions (back panel positions 68-71). The 4 bit EDGE counter is located in position 21 and the CORNER counter in position 20.

It should be noted that the angle counters have DOWN of the Done card wired to the *EUP on the Register. This is correct since the UP and DOWN signals from the Done card are high signals and a low signal is needed to enable the desired count.

4.3.7 I/O Directive Paths

Directive data is brought into or transferred out of the Clipper via directive cables. Most of the directive bits are wired to the Directive card which accepts and registers the directive data. The directive cable contents and bit assignements are shown in the following table. The card names in parenthesis indicate the card where the directive terminates or originates.

Cable Pins	Function	
1	Unused	
3	DIR (1)	(Directive)
5	DIR (2)	"
7	DIR (3)	"
9	DIR (4)	"
11	DIR (13)	"
13	DIR (9)	"
15	DIR (11)	"
17	DIR (10)	"

Cable Pin	Function	
19	DIR (5)	(Directive)
21	DIR (6)	"
23	DIR (7)	"
25	DIR (8)	"
27	DIR (17)	(SLAVE)
29	Unused	
31	"	
33	"	
35	"	
37	DIR (16)	(Directive)
39	DIR (18)	(ADAGE)
41	DIR (19)	"
43	DIR (20)	"
45	DIR (21)	"
47	DIR (22)	"
49	DIR (23), DIR (12)	(ADAGE) (Directive)
51	DIR (24)	(ADAGE)
53	Unused	
55	DIR (25)	(ADAGE)
57	DIR (14)	(Directive)
59	DIR (15)	"
61	DIR (26)	(ADAGE)
63	Unused	
65	"	
67	SWAP	(Directive)

Cable Pin	Function
69	WAITING INPUT (ROM)
71	CLRHIT (GROUND)
73	SETHIT (DONE)
75	CLRAIC (ROM)
77	SETAIC (DONE)
79	Unused
81	Master Clear from PDP-10
83	ACK FROM CHN (CLOCK)
85	CLIPPER OUT FLAG (CLOCK)
87	TUT TUT FORBID (SLAVE)
89	MD (2) (Directive)
91	MD (1) (Directive)
93	ACK TO CHN (CLOCK)
95	CLIPPER IN FLAG (Directive)
97	CLIPPER CLOCK (CLOCK)

Figure 4.5

4.3.8 Directive Control of the Program Counter

The algorithm for the Clipping Divider is implemented by driving specific words of Read Only Memory which turn enable groups of bits that control the function of the Clipper. Only one ROM word line is driven at a time. The address that selects the ROM word to be driven and thus the state of the machine originates in the program counter which is located on the Microcode Control card. This program counter is conditioned by the directive data that has been accepted by the Clipper, the state of the data within the accumulators, and the previous steps executed.

The Small Binary Number card is an interface between the Directive and Microcode control. The Small Binary Numbers card sends dispatch offsets to the Microcode Control card. The following signals are wired from the Directive to the Small Numbers card:

WIDE
LONG
*EVEN
*SETPT
ABSOLUTE
DOT
*MORE
*LINE
*ODD
*LOAD
SIZE
RELATIVE
SELF
*BOX

Two flip flops on the Microcode Control card have been assigned the function AGREE and CUR. The signals *CUR,AGREE and *AGREE needed for the Small Binary Numbers card are thus wired from the Microcode Control card to the respective inputs of the Small Numbers card. The dispatch selection signals for the Small Numbers card are supplied by three wires from the ROM (BRADD(1)-BRADD(3)). Note that BRADD(0) is tied to ground. The Small Numbers card generates dispatch offsets. This data is contained in the *EXAD outputs which are wired to the *EXAD signals of the Microcode Control. The *EXAD bits 1-5 on the Microcode Control card are tied to Vcc.

The CUR and AGREE flip flops are cleared by ROM signals.

Other ROM signals allow setting of the Done card signals dictate that the flip flops should be set. The Microcode Control card contains test logic that tests for certain states that condition the updating of the program counter. The signals that determine which of the tests should be made are ROM signals which are tied to the 8 CI inputs of the Microcode Control card. Test numbers ans their meanings are shown below.

T Tests

TT	200	LOAD (Directive)	FETCH (Directive)	ZP-(SB-PX)	ZN-(SB-NX)
	100	BOX (Directive)	ATOM (ADAGE)	PTOM(ADAGE)	ZTOS(ADAGE)
	40	LONG (Directive)	EVEN (Directive)	STOM(ADAGE)	STOS(ADAGE)
	20	3 D (ADAGE)	LINE (Directive)	*PTOM(ADAGE)	DOT(Directive
		10	4	2	1

U Tests

UT	200	Clip Reject	(DONE)
	100	Trivial Reject	(DONE)
	40	Not Used	
	20	Not Used	

The D signal needed by the Microcode Control is wired from the DONE signals on the Done card. *D is obtained by wiring the D signal to the INVL input on the Microcode Control which inverts the signal. The output of the inverter is fed into the *D signal. The Small Numbers card also contains a one shot (mono-stable multivibrator) which provides system master clear. This one shot is set of by the master clear from the driving computer or the master clear switch from the control panel. System MASTER CLEAR clears the program counter through the wires which feed the CLEARP input on the Microcode Control card. The Adage unit is also reset by the MASTER CLEAR signal on the Adage Interface card.

The ADDR address bits on the Microcode Control card are fed by the ROM signals. The YES, NO, SEL, and PUSHR signals are also wired from the ROM cards. The value of the program counter is decoded and wired to the ROM driver and the Clock cards.

4.3.8 ROM Bit Selection

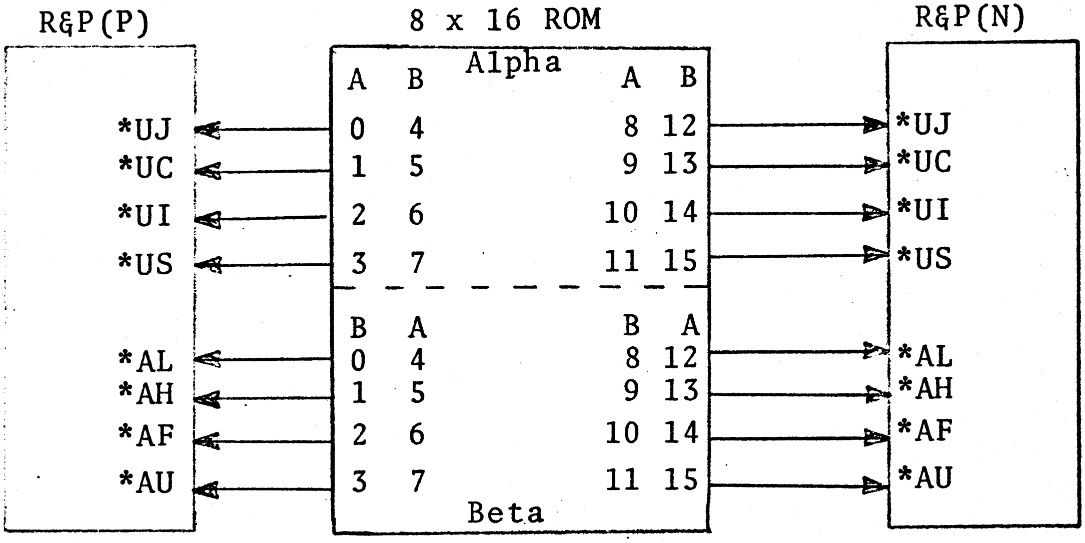
The ROM Driver, which selects and drives a ROM word, is fed address data from the program counter on the Microcode Control card. This address information is contained on the ROMAS lines which are wired from the program counter outputs to the ROM Driver card.

The ROM Driver outputs are wired in parallel to the six ROM cards. The selected ROM word enables a group of ROM bits which are sent throughout the Clipper for control.

4.3.9 Poke and Selection Control

The selection of the input to the upper and lower register of the accumulator (R & P card) is controlled by two of the four sections of the QUAD 8 x 16 ROM. The Alpha ROM bits drive the upper bits in the order shown in Figure 4.6. The switches that control the output of the accumulators on the R & P card are controlled by the ROM bits from the Beta side

Upper and Sum Switch Control
(Same order for both A&B sides)



Register Input Selection Switches
(Output switches for R&P cards)

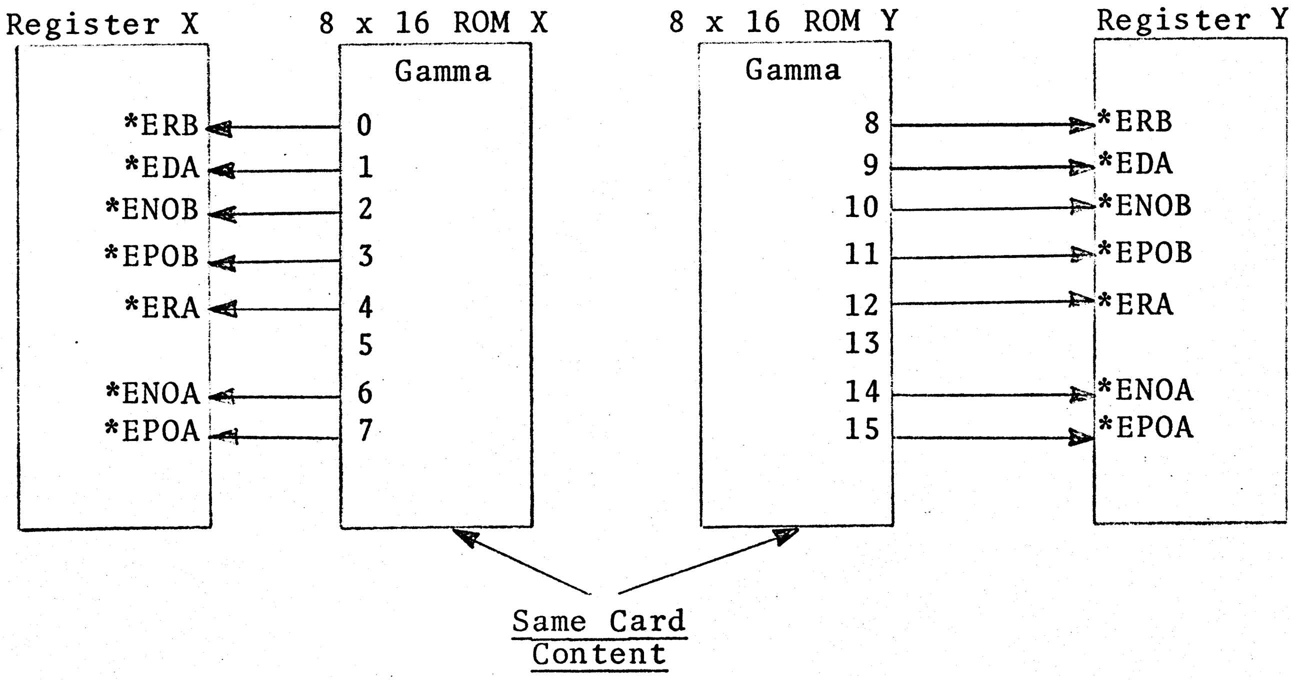


Figure 4.6 (continued)

of the card. Again, the order is shown in Figure 4.6. The input selection switches for the storage registers on the Register card are controlled by the Gamma side of the 8 x 16 ROM. Both the X and Y coordinate 8 x 16 ROMs have the same content, but in storage register selection control, Gamma bits 0-7 are wired to the selection switches on the X series Register cards and bits 8-15 are wired on the Y coordinate 8 x 16 ROM to the selection switches on the Y series Register cards. The order of bit wiring is shown on Figure 4.6 below.

Poke and Lower Switch Control

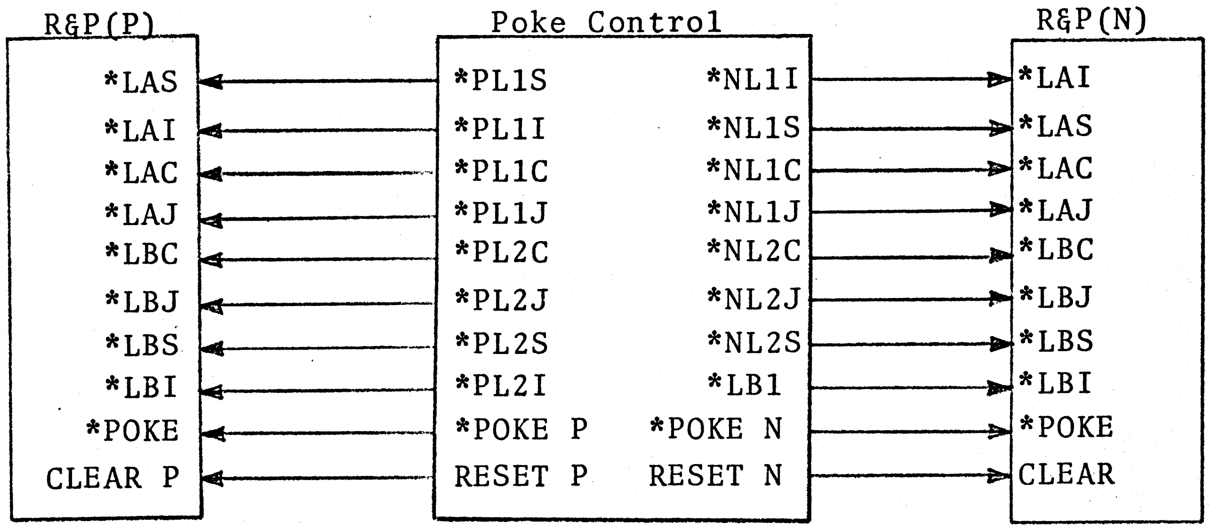


Figure 4.6

It is possible to send either the true or the compliment form of the data in the storage registers of the Register card to the accumulators on the R & P card. Output switches that allow this option are controlled by the ROM signals. But since there is one stage of inversion between the Register card output and the registering of data in the working registers of the accumulator, the selection is inverted from what it at first glance appears to be. When *ECOMP is low, the true value of the registers is selected and put on the output bus so that data registered into the accumulators is in compliment form.

The Poke Control card controls the poke enable and clear signals sent to the adders on the R & P card and the selection switches to the lower registers on the R & P card. The wiring connections are shown in Figure 4.6. The POKE signals for the P side accumulators are *PL1 for the A side of the card and *PL2 for the B side of the card. Similarly, *NL1 provides the POKE signals for the A side of the N series R & P cards and *NL2 is wired to the B side.

These signals from the Poke Control card are dependent on a small 8 X 16 ROM located on the Poke Control card. The word lines are selected by a 1 of 8 decoder controlled by LE(1)-LE(3) from the 64 Word ROM. Logic fed by sign and likeness data from the Sign and Carry card also condition these selection signals.

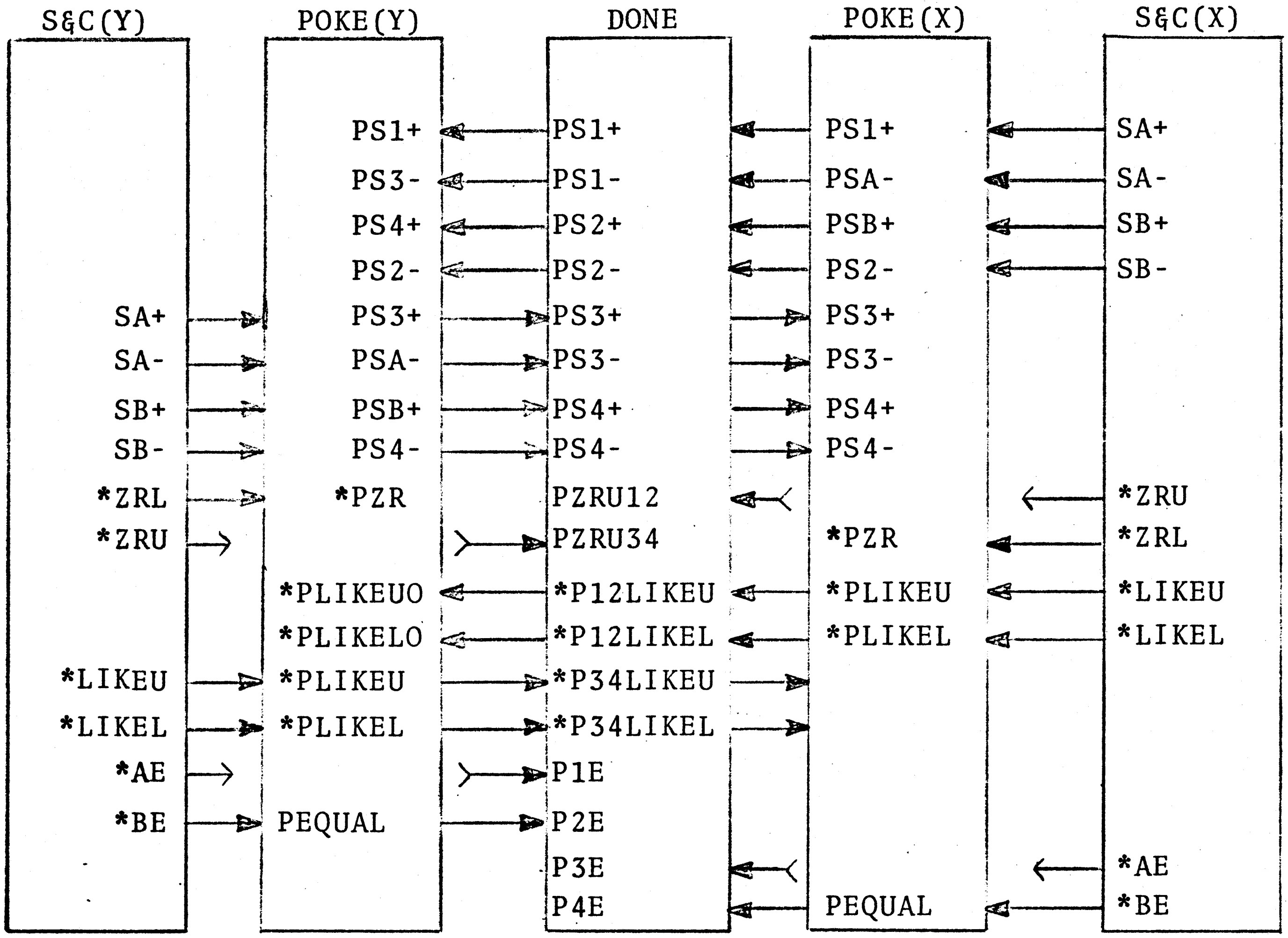
This same logic and an 8 x 8 ROM on the Poke Control card are used to develop the poke generating signals needed by the R & P accumulators. *POKE PUA drives *POKE UA on the P series R & P cards and so forth. The 8 x 8 ROM is driven by a decoder working on the 3 bit binary code PE(1)-PE(3) from the 64 Word ROM.

The CLEAR signals of the R & P are wired from the RESET signals on the Poke Control card. These clearing signals on the Poke Control card are in part controlled by the CLEARPU, CLEARPL, CLEARNU. CLEARNL signals from the 64 Word ROM.

The shift signals on the R & P card are tied to Vcc as shifting is not used in the Clipping Divider. Certain steps of the algorithm require special poking. For this reason, ROM state signals CLIP, DIVIDE, BOX and EDGE are wired to the Poke Control card to be used in poke generating logic. Special poking is also required in boxing. The relevant information is fed to the Poke Control card by ROM signals ONLY OFF, ONLY ON, ONLY REV.

4.3.10 Done Conditions

The Clipping Divider contains logic which allows the determination of the completion of the clipping and division processes. As long as these operations are still in process, they are controlled by the signals from the Poke Control card. These signals are functions of the read only memory on the Poke Control card (as noted below) and of signals from the Sign and Carry card. The main paths are shown in Figure 4.7 below.



Done Conditions

Figure 4.7

The signals to each of the Poke Control cards (X and Y) come from both the X and Y coordinate Sign and Carry cards. The PS1+,PS2- inputs to both Poke Control cards from the X series Sign and Carry cards. PS3+ and PS4- come from the Y series. The PSA- and PSB+ inputs come from the Sign and Carry card signals SA- and SB+ from teh S & C cards on the same coordinate. PS3- and PS4+ come from the SA- and SB+ outputs from the S & C cards for the other coordinate.

The signals *PLIKEU and *PLIKEL are wired from the likeness signals on the S & C card in the same cage while *PLIKEUO and *PLIKELO are wired from the S & C card in the opposite cage. The zero (ZR) and equality (E) signals are wired as shown Figure 4.7

The Signa dn Carry card signals are also wired to the Done card to allow the determination of when to stop a clipping or division process. Sign, likeness and zero state signals are wired from the Sign and Carry cards to the Done card. Signals from the different quadrants are denoted on the Done card by numbers as follows:

1= A side of X
2= B side of X
3= A side of Y
4= B side of Y

Although the AD signals from the high order bit R & P card are wired into the Done card, the *SHIFT signals developed are not used.

The signals *PTHRUC and *NTHRUC are wired to the Poke

4.4 The Foster Idiosyncrasies

"The OA output of the previous point R & P cards should be wired to the IA inputs of the new point R & P cards then down to the register inputs POA. The OA output of the new point R & P cards should be wired to the IA inputs of the previous point R & P cards and then down to the Register inputs NOA. However, due to a minor (?) inaccuracy in specifying the back panel wirelist, LDS-1 has an idosyncrasy (just one?). The register NOA and POA inputs are reversed. Therefore, the previous points output goes to NOA and the new point output goes to POA (similarly for NOB and POB). This was remedied by interchanging the selection wires *ENOA and *EPOA. The B side wiring is the same as the A side. This connection allows the output of the R & P cards to be stored in any selected register or to be placed into the output bus."

"The upper and output switches of the R & P cards are controlled by the Small ROM card. The alpha ROM bits 0-7 drive the upper switches of the previous side in the order joint bus, compliment bus, input bus, sum bus for the B side followed by the same selection for the A side. Bits 8-15 control the next point side in the order compliment bus, joint bus (another idosyncrasy, huh?), input bus and sum bus for B and joint bus, compliment bus, input bus and sum bus for the A side."

RATIO AND PROPORTION CARD

5.1 Objective

The Ratio and Proportion card is the main arithmetic element in the Clipping Divider. It is designed to serve as the accumulator in the midpoint algorithm and the other functions of the clipper. Each card has two separate 4-bit mid-point units which are intended to perform two separate additions simultaneously.

The Clipping Divider employs four groups of Ratio and Proportion cards. Each group is comprised of six cards to allow the manipulation of numbers up to 24 bits long.

5.2 Reference

E & S logic drawing No. 101105-600, Ratio and Proportion Card.

E & S data path drawing No. 101105-900, Ratio and Proportion Card Data Paths.

E & S block diagram No. 101123-900, Clipping Divider Registers and Busses.

5.3 Card Contents and Function

The two symetrical halves of the R & P card called the "A" and "B" sides each contain a 4-bit adder, two associated shifting registers (referred to as "working" registers), and input and output switches. The registers are labelled as to their position on the logic drawing. Thus, RUA means Register Upper A side, RLB means Register Lower B side, etc. The bits within the register are subscripted with the subscript of 1 indicating the most significant bit.

The basic data flow begins at the card input which feeds 4-way selection switches. The selected input then goes to the registers. The output of the register is used to project the carry state of the adders. Register output is then sent to the adders. The adder output is sent to 4-way output selection switches which select the full sum, half the sum, or the output of either the upper or lower registers from the opposite side of the card.

The details below will follow the order of data flow as indicated above with interjections concerning control signals. As the card is almost the same on both sides, only one side will be described, except where the sides differ.

5.3.1

There are four groups of possible inputs to the R & P registers:

1. External data input from the opposite group of R & P cards (i.e. P for N and N for P). This input is designated as "I" for the lower registers and "D" for the upper registers. IA(1) is, then, the most significant bit of input for register RLA.

2. Joint input busses from the register cards feed the lower register on one side of the logic drawing and the upper

register on the other side. These busses carry the designation of the lower register they feed. (Thus, RA(1) feeds RLA and RUB.)

3. The output of the adder output selection switch is used to put full sums or half sums of the two registers associated with an adder back into the registers themselves. Alternately, if output of the uppor or lower registers from the opposite side of the card was selected, it is possible to interchange upper or lower registers (e.g. RUA with RUB).

4. The compliment of the registers themselves can serve as input. This is used in the formation of differences.

Associated with each bit of the registers is an input switch (4-wide AND-OR-Invert gate) which allows the selection of the desired input data to the register. The selection for the lower registers is determined by enabling signals from the Poke Control cards. The selection for the upper register comes from the Small ROM cards. All of the switches on the card invert the data signals going through them. If data in the registers is to be thought of as true, then data inputs and outputs on DA, RA, RB, IA, IB, SA and SB must be thought of as inverted. The selection lines for data input contain inverters so that a low signal to the card is required to select the desired input. (If two inputs are selected simultaneously, the switches will perform the logical OR function of the outputs, which is not useful.) If no input bus is selected, the switch will output a value of minus zero (all 1's).

5.3.2 The Registers

The selected input is clocked into the registers. The incoming clock signal is normally high and becomes low for a short interval called "clock time." Whether or not the clock pulses place data in the register is individually controlled for each register. The clock is ANDed with conditioning signals, called "pokes," which originate in the Poke Control card and are designated by their register names.

The individual registers are capable of shifting data to the left. This capability is included so that increased precision can be obtained if the numbers in the register are small enough to be doubled. Shifting is controlled by lines *SHIFTA and SHIFTB. As no shifting is required on the B side, *SHIFTB is tied to Vcc. Shifting is used for doubling to insure that the division process will not be terminated until a sufficiently accurate answer has been obtained. The poke signals take priority over the shift signals. Thus if the full sum is poked back into a register, it will not shift, but if the register is not poked, shifting will occur.

The registers may also be asynchronously cleared by the CLEARU and CLEARL inputs from the Poke Control cards. These clears take precedence over the shift and poke commands. The register is cleared immediately after the clear command is given.

5.3.3 Carry and Equality Functions

The output of the R & P registers is sent to the combinatorial logic shown on sheets 2 and 3 of the logic drawing. This logic is used to generate four functions. The first two have to do with the projected carry state of the adder. As the "carry in - carry out" sequence of the adders is too slow, and the sign of the sum needs to be known to determine what to do with the sum when it is finally computed, seperate "sign and carry" logic has been designed to perform this task. This logic is on the Sign and Carry cards (described in section 6). The data needed by the Sign and Carry card is developed by the carry logic on the R & P card. AND-OR logic is used to compare the outputs of the upper and lower registers. For example, the upper AND gate shown on the logic drawing in position 19 checks to see if both most significant bits are 1, in which case a carry will be generated. The lower AND gate in position 19 goes high if the most significant bits are both 1, and so forth. The two signals developed are "not carry generate" (*AG (1-4) and "not carry kill" (*AK (1-4). In the "carry propagate" case, (i.e. the numbers being added are exact compliments) both signals will be high.

In addition to the carry function, a check is made to see if both numbers being added are equal. The exclusive OR circuitry shown on the logic drawing performs this task. The AE (C12) signal is used by the Done card to determine when clipping or dividing can be terminated.

Logic shown on sheets 2 and 3 also developes "doublable" functions. These can be used to control shifting of the R & P registers when the numbers in the register are small enough to be doubled.

5.3.4 Adder and Adder Output Switches

The output of the upper and lower registers is then sent to the corresponding adder and summed. Each adder in the Ratio and Proportion card has an associated output switch, (4-wide AND-OR-Invert gate), capable of selecting either the full sum of the adder, half the sum, or the output of the upper or lower registers of the opposite side of the card. Intercard shifting paths AA4, AB4 (outputs) and AAO, ABO (inputs) are provided to pass bits of the sum to the right when the half sum outputs have been selected. The selection of the direct register outputs from the opposite side of the card makes it possible for data to be passed from one coordinate to another within the card. Selection signals for the output switches originate within the Small ROM card. Output is sent to the Register cards and to the opposite group of R & P cards.

SIGN AND CARRY CARD

6.1 Objective

The Sign and Carry card is designed for use in generating sign and carry information for two separate adders. Two high-speed carry chains generate "carry in" signals for two groups of 4-bit adders each up to 24 bits long. Data indicating the signs of the numbers being added are used with carry state data to determine the sign of the sum. This determination is made long before the adders have actually calculated the sum, allowing the sign data to be used in deciding what to do with the sum. The Sign and Carry card also generates several functions needed to do the vector arithmetic required by the Clipping Divider.

6.2. Reference

E & S drawing No. 101106-600, Sign and Carry card.

E & S Block Diagram No. 101106-900, Sign and Carry card Block Diagram.

6.3 Card Contents and Function

Carry generating data is sent from the Ratio and Proportion card for use by the carry chains of the Sign and Carry card. The carry data generated are used with sign data from the R & P adders and registers to generate five special sign related functions. The Sign and Carry card also generates "carry injection" functions used for rounding or injections used in correcting for compliment arithmetic.

The following discussion will deal with the functions of the Sign and Carry card in the order indicated in the previous paragraph.

6.3.1 Carry Chains

The carry chains are designed to provide carry input information for the 4-bit adders of the Ratio and Proportion cards. Each card contains two separate carry chains which are labelled A and B, corresponding to the A and B sides of the R & P card. Each chain is capable of handling an adder up to 24 bits wide. The clipper employs one Sign and Carry card with each of the four groups of Ratio and Proportion cards. The carry chains (sheets 1 and 2 of the logic drawing) are of minimal design and fast. Carry data is sent from the R & P card indicating the respective adder is in one of three states:

1. Carry generate (*AG, *BG)
2. Carry kill (*AK, *BK)
3. Carry propagate (i.e. not carry generate, not carry kill)

The carry information from one stage to the next is contained in four lines which are gated together to detect one of the two conditions which will produce a carry for the appropriate adder. If a "carry generate" is commanded by the Ratio and Proportion card, a carry command will be sent to the adder in question. Alternately, if a propagate condition exists (i.e. not carry generate and not carry kill), and a carry is indicated by the previous stage, a carry signal is also generated. Carry outputs are labelled CIA(4), CIB(4), CIA(8), CIB(8), etc.

The carry chain is extendible for use with adders longer than 24 bits. For this purpose outputs NOA, COA, *AG(1-24) are provided, which should be connected to the next most significant card inputs of NIA, CIA, *AG(21-24) and *AK(21-24) respectively.

6.3.2 Sign Generating Circuits

The sign generating circuits (sheet 3 of the logic drawing) compute the sign outputs for both carry chains. Both true and compliment outputs are generated. Data is gated such that the sum of an adder is declared positive if the numbers present at each of the two associated registers on the Ratio and Proportion card (see section 5) are both positive (A++) or the numbers are not both negative (*A--) and a carry has been generated or propagated by the most significant stage of the carry chain (AG(1-24). The logic equations relating to sign generating functions and also the carry injection functions discussed in section 6.3.4 can be found on sheet 3 of the logic drawing.

6.3.3 Special Sign Functions

The Sign and Carry card generates two groups of signals used by the Poke Control and Done Condition cards to control clipping (sheet 4). These signals are functions of the signs of the R & P registers (UA(1), LA(1), UB(1), LB(1)), the signs of the sum (SA+, SA-, SB+, SB-) and ROM enabling signals (EA+, EA-, EB+, EB-). The signs of the registers are ANDed with the ROM enabling signals. The resultant signal is then ANDed with the sign of the sum of the appropriate adder. The resultant signals are then ORed together. The output indicates that the sign of either the A or B upper register is the same as the sign of the sum, (*LIKEU), or that the sign of either the A or B lower register has the same sign as the sum, (*LIKEL).

The *ZRU and *ZRL signals are derived by a similar masking process. They indicate that an adder is generating zero. In this case, AP 1-24 will be high, indicating the all propagate state. *ZRU will go low if EA+ is high and UA(1) is low (positive) or EA- is high and UA(1) is high (negative). The situation is analogous with *ZRL.

The rejection signal (*REJ) developed by the sign related functions is not used in the Clipping Divider.

These five logic output levels allow the Sign and Carry card to detect sign conditions selectively for doing vector arithmetic. During the clipping process, the enabled lines will be set so that the "out" signs are enabled. Thus, EA- and EB+ will be set. If the data in both upper and lower registers are out, as indicated by their signs, the rejection level will so indicate. Clipping proceeds with either the *LIKEU or *LIKEL signal controlling how the sum is used. Finally, a zero will be detected indicating a point on the edge of the window, and the resulting signal on the *ZRU or

*ZRL line will terminate clipping. During division, only the A unit is enabled. Both EA- and EA+ are turned on, so that the *LIKEL and *LIKEU lines indicate whether the sign of the sum is like the A upper or A lower register. The B unit does no detection because it is being used as a scaling unit. Division terminates either when the A unit generates a zero or when data in the B unit registers are found to be equal.

6.3.4 Carry Injection

The carry injection data generated by the Sign and Carry card is a function of the signs of the R & P adders and "round" and "injection" signals from the Small Binary Numbers card. The signals developed (INA, ICA, etc.) control rounding and injections required by twos compliment arithmetic. If the round signal is high (i.e. *AR is low), carry injection will depend soley on the sign of the sum. A positive sum will not generate an injection, but a negative sum will. Thus, the adder rounds towards zero. Injection is different for the A side and the B side when they are in all propagate conditions. An injection will be sent to the A side adder, while none will be sent to the B. Thus, the A unit will output plus zero and the B unit minus zero.

If rounding is not commanded, injection depends soley on the injection signal (*AI). Injection occurs if commanded.

6.3.5 Equality Circuits

In order to test equality of the upper and lower registers, a wide AND gate is provided (sheet 6) for each side (A and B). These gates combine the equality signals from the individual 4-bit R & P cards into total equality signals.

REGISTER CARD

7.1 Objective

Register cards are used to make up the basic storage registers of both the Clipping Divider and the Channel Control. Each card represents a 4-bit slice of data. Each of the two similar halves (A and B) contain up to 8 individual registers. In addition, the A side contains a counter, and the B side has gating allowing it to accept special input data.

7.2 Reference

E & S logic drawing No. 101107-600, Register Card complete.

E & S logic drawing No. 101107-601, Register Card 2 x 5 Bit.

E & S logic drawing No. 101107-602, Register Card 2 x 8 Bit.

E & S block diagram No. 101123-900, Clipping Divider Registers and Busses.

7.3 Card Contents and Function

There are four possible input busses for the registers. One of the four is selected by four-way switches and clocked into the registers selected by a decoder. Register output is also enabled by an appropriate signal from a decoder. The output is inverted and passed to output switches which select either the true or compliment output. The selected input is also immediately available as output without having been registered. When appropriately enabled, the selected input to the A registers is sent to a counting register and on to an adder which will count the data up or down depending on an external command.

The following description will follow the basic data path from the input to output and then describe the counter and the special B side input. Reference should be made to figure 7.1 and the logic drawings.

7.3.1 Input

There are four basic input busses coming to the Register card:

1. The IN bus, which is four bits wide and carries data in true form from the Matrix Multiplier or the Channel Control (depending on the system configuration, is available to both sides of the card.

2. The INA, INB bus also carries true data from either the Matrix Multiplier or the Channel Control and feeds the A and B sides respectively.

Either the IN or INA, INB bus is selected by a two-way switch (2-wide AND-OR-Invert gate). Enabling signals *EINB and *EINA, which come from the Directive card, determine selection. The output of the two-way switches forms bus DA, DB which is available as immediate card output and also sent to the register input selection switches. The bus DA, DB is in compliment form.

3. The POA, POB bus contains data in compliment form generated by the P series R & P cards.

4. The NOA, NOB bus carries compliment data generated by the N series R & P cards.

In addition, the output of the registers themselves can be recycled as input for the register input switches. This data is inverted by the output selection switches and, thus, sent to the input switches in compliment form.

7.3.2 Register Input Switches

These switches are made up of 4-wide AND-OR-Invert gates. The switches select between 1. the DA, DB bus, 2. the POA, POB bus, 3. the NOA, NOB bus, or 4. the output of the registers as input for the 4-bit registers. The ROM 8 x 16 generates the signals that determine this selection. These signals are called *EPOA, *EPOB, etc., and are sent to both sides of the card. The selected input is inverted by the switches and becomes BUS1A, BUS1B, which is now in the true form and sent to the registers.

7.3.3 Register Selection and Function

The data of BUS1A, BUS1B are registered in only one of the registers on each side of the card. Selection is determined by the 4-bit binary number CODEOA - CODE3A, which is generated by the Directive card. This number is decoded and the output ANDed with the clock pulse. The resultant signal clocks information into the selected registers. The registers are SM63's (or equivalents) which do not propagate their contents until enabled by an output enabling signal. The binary number CODE4A - CODE7A, also from the directive, is decoded and enables the output of one of the registers. The register selection signals (both input and output) are sent to both sides of the card. For example, an input selection code of 0011 would clock data into register 3 (fourth from the top of the logic diagram) on both sides of the card. Similarly, an output selection code of 0010 would select the output of register 2 on both sides of the card. Either input or output selection codes equal to or larger than 1000 (i.e. the most significant bit is high) will select no register. The outputs of all of the registers are tied together to form a common bus (BUS2A, BUS2B). Discrete pull up resistors and pull up resistors within the Register's IC's provide the necessary resistance so that register output will be in correct form. If no register output has been selected, BUS2A, BUS2B will contain all 1's.

The data on BUS2A, BUS2B is inverted and both the true and compliment information is presented to the output switches.

7.3.3 Output Switches

The output switches consist of 2-wide AND-OR-Invert gates. The ROM signals *ECOMP and *ETRU select either true or compliment data from BUS2A, BUS2B. The selection of true data means that if this data is recycled into the registers, it will be registered in true form and not that the output busses contain true information.

7.3.4 Output

There are three basic output busses:

1. The OUTA, OUTB bus is taken directly from the OUT2A, OUT2B bus after inversions for buffering. OUT2A is sent to the Adage Interface card and OUT2B is sent to the Slave Control card. This bus contains all 1's if no input has been selected.

2. The DA, DB busses contain the compliment of data contained in either the IN bus or the INA, INB busses. DA is sent to the P series of the R & P cards and DB is sent to the N series. Again, if no input has been selected, the bus contains all 1's.

3. The data selected by the register output switches form the RA, RB busses. RA is sent to the P series of the R & P cards and the RB to the N series. The busses contain all 1's if neither true nor compliment outputs have been selected.

7.3.5 Counter

The counter on the Register card is designed to count either up or down under external command. The counter circuit provides sufficient logic so that the counters can be cascaded into up-down counters longer than four bits. When the units are cascaded, the carry chain is implemented with a single "AND" gate delay per 4-bit section. There are two carry chains, one for up counting and one for down counting. The counting itself is performed by using a 4-bit adder circuit.

The counter logic is shown on sheet 2 of the logic diagram. Selection switches (2-wide AND-OR-Invert gates) choose between 1. data from BUS1A (i.e. from the register input bus), and 2. the recycled data from the count adder. A low signal *E BUS COUNT from the Directive selects the data on the BUS1A and clocks it into the double rank count register. A high signal on *E BUS COUNT selects the output of the count adder. This data is clocked into the register when a Done card signal called COUNT is high and either CARRY IN UP A or CARRY IN DN A is high. The data on BUS1A is true; the data in the register is, thus, in compliment form. Output from the register is sent back through the open collector NAND gates and to the register output BUS2A when enabled by the Directive card signal *E COUNT OUT. Count register output is also sent to a 4-input AND gate which detects the all zero case and generates the signal ZERO OUT A. If this signal is high and if CARRY IN DN is high, the signal CARRY OUT DN A is also high. The count register output, being in compliment form,is inverted and then passed on to a 4-input AND gate which detects the all 1's case and generates the carry signal CARRY OUT UP A if CARRY IN UP A is high. The CARRY OUT signals are used to generate borrows and carries when more than one card is used to make a counter more than 4-bits wide. When only a 4-bit counter is needed, or in the case of least significant counter, CARRY IN UP A should be tied to Vcc. The 4 bits of count register output, now in true form, are sent to the count adder. The adder adds either -1 (1111) or +1 (by initiating a carry). Adder output is recycled through the selection switches and the open collector NAND gate and available on the BUS2A in true form when the enabling signal *E COUNT OUT is low.

7.3.6 Special Inputs for the B Side

When the enabling signal *E WIRE from the Directive is low, which happens at the same time CODE4A is high (none of the register outputs are enabled), 4 bits of input data from the Slave Control cards are sent to the BUS2B through an open collector NAND gate. This data is then available in true or compliment form on the RB output bus.

7.3.7

The Register card is designed to accept either true or inverted clock pulse. If a compliment clock is available, it is wired to C28, inverted, and jumpered on the back panel from C30 to C32.

7.4 Special Applications in the LDS-1 Clipping Divider

Several versions of the Register card exist, differing only in the number of components actually in place.

The Clipping Divider of LDS-1 employs two groups of Register cards (X and Y). Each group is made up of five cards and each card contains five registers[1]. The Register cards, thus, have the capacity of handling two groups of five words of data where each word is up to twenty bits long.

In the Y series of Register cards, four 4-bit counters are used as angle counters (Sec. 13.3.1). Each counter operates independent of the others. Control signals come from the Done card.

In the X series of Register cards, two 4-bit counters and one 8-bit counter are used. These counters only count up (i.e. *EUP is tied to ground and *EDN to Vcc). Counters on the Register cards in back panel positions 18 and 19 form the 8-bit counter. In this case, CARRY OUT UP A (C16) of card 18 is connected to CARRY IN UP A (C5) of card 19 and CARRY OUT DN A (C94) is connected to CARRY IN DN A (C4). In all other cases, CARRY IN is tied to Vcc.

[1] E & S logic drawing No. 101107-601. The names of the registers, which relate to the functions defined in the algorithm (section 3) are given on E & S block diagram No. 101123-900, Clipping Divider, Registers and Busses.

POKE CONTROL CARD

8.1 Objective

The Poke Control card functions in conjunction with the Ratio and Proportion and Sign and Carry cards to control the loading and clearing of the R & P registers necessary for clipping and division and the other functions of the Clipping Divider.

8.2 Reference

E & S logic drawing No. 101108-600, Poke Control Card.

E & S block diagram No. 101108-900, Poke Control Card Block Diagram.

8.3 Card Contents and Function

The Poke Control card generates clear, poke, and enable signals for the R & P registers. The clear signals are dependent on input commands from the ROM and Directive cards and sign information from the Sign and Carry card. These signals also condition a "poke inhibit" signal, which is used along with more sign information and signals from an 8 x 8 Read Only Memory to develop poke signals that control the loading of the R & P registers. The enabling signals that select the input for the lower R & P registers are developed by an 8 x 16 Read Only Memory and conditioned by sign data. The Poke Control card also distributes its input clock pulses to other cards in the system. The following description will examine these functions in detail in the order listed.

8.3.1 Clear Signals

A NAND gate, used to perform the logical AND function, controls the clear signals (RESET NL, RESET NU, RESET PL, RESET PU). Thus, there are two groups of conditioning signals that must be in appropriate states in order to generate a clear. For the purpose of the following discussion, these groups will be arbitrarily labelled group 1 and group 2. Group 1 contains sign information (NSA-, NSB+, etc.) that is clocked into a single rank D-type latch when the ROM signal EDGE is high. Output from the latch is gated together and conditioned by ROM signals having to do with the boxing algorithm. (ONLY OFF, ONLY ON, ONLY REV). The results determine the state of group 1. The tables below show how this circuitry functions.

	++	+-	-+	--
none	X	X	X	X
only on		X		
only off	X		X	X
only rev	X	X		

P series

	++	+-	-+	--
none	X	X	X	X
only on		X		
only off	X		X	X
only rev				X

N series

X = condition satisfied

Group 2 includes the ROM signal CLEAR and Directive card commands FROM, ABSOLUTE, INPUT AND *CS. The following table shows the appropriate combinations of these signals.

P lower	P upper	N lower	N upper
CLEAR PL or FROM ABSOLUTE INPUT & *CS	CLEAR PU or *FROM INPUT & CS	CLEAR NL or *FROM ABSOLUTE INPUT & *CS	CLEAR NU or FROM INPUT & *CS

A poke inhibit signal (*INH) will be generated unless the first group conditions are satisfied, the second group conditions are not satisfied, and the relevant inhibit command from the Directive card (INHIB A or INHIB B) is low. This inhibiting signal is used as a conditioning signal for the poke commands.

8.3.2 Poke Commands

The poke commands are developed by 4 input NAND gates. The four required conditions are satisfied when: 1. the appropriate bit from an 8 x 8 Read Only Memory is selected, the bit selection is determined by a decoder which decodes a three-bit command from the ROM (PE), 2. poke inhibit has not been commanded (i.e. *INH NLA, etc. is high), 3. the ROM signal CLIP is high or the right combination of sign or likeness information from the Sign and Carry card is available, 4. the ROM signal DIVIDE is high and appropriate likeness or equality information for the Sign and Carry card is available. The combinations that satisfy conditions 3 and 4 are shown in the tables below.

Assuming conditions 1 and 2, poke unless the following signals are high:

	Lowers	Uppers
Group 3.	CLIP	CLIP
	*LIKEL	*LIKEU
	*LIKELO	*LIKEUO
	*+-+-	*THRUC
Group 4.	DIVIDE	DIVIDE
	*LIKEL	*LIKEU
	*BEQUAL	*BEQUAL
	*ZR	*ZR

8.3.3 Enable Signals

The Poke Control card also develops the signals that select the input for the lower registers of the R & P card. The signals *NL1C, *NL1J, *NL2C, *NL2J (with a parallel P series) are simply inverted output of an 8 x 16 Read Only Memory fed by a decoder that decodes a 3-bit code from the ROM (LE). The other enabling signals are conditoned by the ROM signal EDGE. If EDGE is low, the signals *NL1I, *NL1S, *NL2I, *NL2S are also simply inverted output of the 8 x 16 Read Only Memory. If EDGE is high, these signals are dependent on data indicating the sign of the R & P adders and an enabling ROM signal BOX. The tables below show how this circuitry functions.

CONDITIONS	RESULTS
1. EDGE low	ROM bit states determine states of *L1S, *L2S, *L1J, *L2J.
2. EDGE high AND BOX low	
a. NSA- low AND NSB+ low AND NS3- low AND NS4+ low	*L1S, *L2S low
b. NSA- high OR NSB+ high OR NS3- high OR NS4+ high OR	*L1S, *L2J low
3. EDGE high AND BOX high	
a. NSA- low AND NSB+ low	*L1S, *L2S low
b. NSA- high OR NSB+ high	*L1J, *L2J low

8.3.4 Clock Distribution

The Poke Control card contains circuitry which takes MASTER CLOCK, inverts it twice and distributes it to other cards in the system.

QUAD 8 X 16 READ ONLY MEMORY

9.1 Objective

The Quad 8 x 16 ROM is made up of four separate 8 x 16 Read Only Memories. These are designed as control circuitry for use in different parts of the system.

9.2 Reference

E & S logic drawing No. 101109-600, Quad 8 x 16 ROM.

Bit specifications: Appendix, Section 19.4

9.3 Card Contents and Function

The four sections of the card are labelled Alpha, Beta, Delta, and Gamma. Only the Alpha section is shown in full on the logic drawing as the sections differ only in placement on the printed circuit card. The following description is applicable to any of the sections.

A four-bit binary code coming from the 64-word ROM is decoded and the output is fed onto one of eight parallel word lines, which are layed out on the solder side of the printed circuit board. Perpendicular to the word lines, and layed out on the component side of the board, are 16 "bundles" of bit lines. Each "bundle" contains four separate lines to allow up to four different words to select one bit. Bit selection is determined by plated holes drilled at intersections of word and bit lines. The four bundled lines are logically

ORed together and the output sent to a connector.

The OR function can be performed either by NAND gates (AND-Invert) or by NAND gates. If NAND gates are used, the output goes high whenever a feeding bit line is activated. If AND gates are used, the output will be low when one of the associated bit lines is activated.

Unused bit lines are left open. By the nature of TTL circuitry, unused inputs float high. Since a low signal activates the gate, high lines have no effect on their gates.

The basic circuit of the 8 x 16 ROM is shown in figure 9.1 below

8 x 16 ROM: Basic Layout

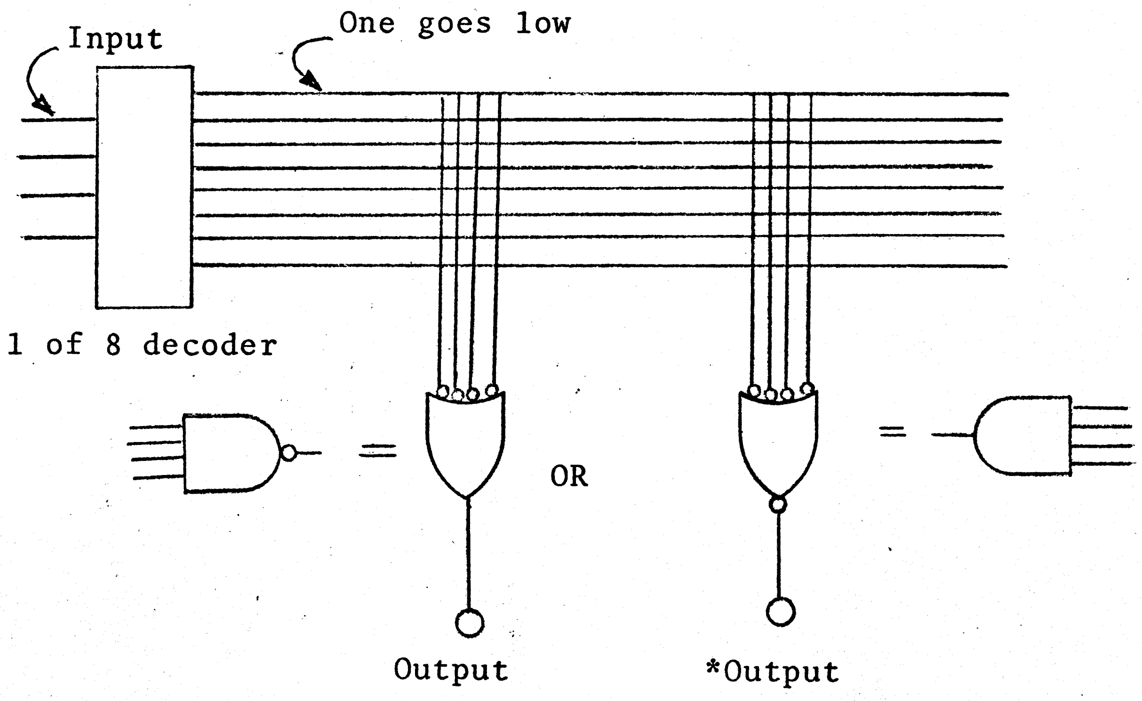


Figure 9.1

9.4 Use in the Clipping Divider

The DELTA portion of the QUAD 8 x 16 ROM is not used in the Clipper. NAND gates are used so that output goes high when a bit is selected. The content of the 8 x 16 ROM is given in section 19.4 of the appendix. The drilling pattern that results in this content is shown in the appendix, section 19.5

In the Clipping Divider, two 8 x 16 ROM's (X and Y) are used to control register loading and output selection in the Ratio and Proportion and Register cards.

64-WORD READ ONLY MEMORY

10.1 Objective

The 64-word Read Only Memory (ROM) cards are designed to provide a very fast read only memory which sends control and enabling signals to various parts of a system.

10.2 Reference

E & S logic drawing No. 101110-600, 64-Word Read Only Memory.

Bit specifications: Appendix, Section 19.5.

10.3 Card Contents and Function

The ROM contains 64 word lines and 144 bit lines which are gated together to form output signals. The basic layout on the printed circuit board will be discussed in conjunction with the selection paths and then component placement, and bit bundling will be explained as it deals with the specific content of the ROM cards.

10.3.1 Layout and Selection Paths

The printed circuit board is layed out with 64 word lines running vertically on the component side. Adjacent to these lines is a Vcc line to which unused circuit inputs are tied. A low signal from the ROM Driver card selects one of these word lines. If no word line is selected, the word lines contain all 1's.

On the solder side of the board, 144 bit lines run perpendicular to the word lines. Where a certain word should select a specified bit line, a hole is drilled at the intersection of the two lines and plated through. These bit lines form "bundles" that make up the input of logic gates. The lines within a bundle are logically ORed together, which allows the selection of any specified output bit by several different words. The number of bit lines in any specified bundle is determined by the number of words that should select the associated output bit. To avoid sneak paths, only one hole may be drilled in any one of the 144 bit lines.

There is a column of integrated circuits on either side of the ROM. Horizontal bit lines, alternating in groups of four, feed a gate on one side and then a gate on the other side. This alternating pattern must be taken into account in the drilling of the holes.

On either side of the board there is a patchwork of horizontal and vertical lines which can be used as connection for an "extender" or for bringing the output of the gates on the upper portion of the card to output connectors. If one of the vertical lines is used as an "extender" it will have two holes, if used to bring output to the connector, the line will contain only one hole. "Extenders" are explained in the next section.

10.3.2 Component Placement and Bit Selection

Quad 2 input NAND or dual 4 input NAND integrated circuits are used depending on the number of bit lines in the bundles involved. Dual 4 input AND gates serve as extenders if more than 4-bit lines are required for a bundle. The output of a 4 input AND provides one of the inputs for a 4 input NAND and thus allows up to seven bit lines to condition an output bit. The 2 input NAND gates may be used only in the lower two IC positions on each side of the board because only these positions have output pins 3 and 11 wired. If outputs 3 and 11 are to be used in these positions, appropriate holes must be drilled in the patchworks on the edge of the card to bring the output to a connector.

The original holes were drilled by an automatic milling machine running off a numerical data tape prepared by a PDP-9 program. Changes require drilling out the old hole and drilling and plating a new hole. The bit specifications and component placements are indicated on the build sheets of the different ROM boards. The boards are numbered historically starting with 01. The two-digit assignment code is printed as the last two digits of the dash number on top of the board. In addition, these numbers are drilled into the board in the following manner:

```
x t t t t t t t t t t x
x u u u u u u u u u u x
```

where x represents a hole always drilled and the tens digit is drilled in one of the positions marked t and the units are drilled in one of the positions marked u. The holes are read starting near the outside edge and counting towards the center of the board.

10.4 Use in the LDS-1 Clipping Divider

The word lines in the ROM correspond to the steps in the algorithm. Five ROM cards are used in the Clipping Divider. The word lines are wired in parallel, so that at any time, the same word line on all of the six cards is enabled.

ROM signals are sent to various parts of the clipper. The signals are indicated as circles on the back panel block diagrams (101123-300).

The actual ROM bits are shown in Section 19.5 of the appendix. The drilling and component placement is shown on the PDP-9 printouts that make up Section 19.6 of the appendix.

ROM DRIVER

11.1 Objective

The ROM Driver receives address data from the program counter on the Microcode Control card and with this data selects and drives one of the word lines of the 64-Word ROM.

11.2 Reference

E & S logic drawing No. 101111-600, ROM Driver.

11.3 Card Contents and Function

The input data for the ROM Driver comes from an 8-bit ROM address register located on the Microcode card. The upper two bits of address select between different ROM driver cards and are connected to inputs labelled P(1) and P(2). If only one card is used in a system, as is the case in the Clipping Divider, P(1) and P(2) are tied to Vcc. The lower six bits of address are decoded onto twelve lines by the Microcode Control card. Each two bits of the address contained in the ROM address registers is represented by four lines. Thus, there are three groups of four address lines (labelled ROMAS), which make up the input data for the ROM Driver. All of the ROMAS lines are available at test points for ease in checking.

The logic involved in word selection is most easily understood if one considers the box of 64 NAND gates on the logic drawing as a four by four matrix where each element in turn consists of four gates. The address signals ROMAS0-ROMAS3 select one of the four gates in each element of the matrix. The signals ROMAS00-ROMAS30 drive one of the four rows of the matrix and the signals ROMAS000-ROMAS300 select one of the columns of the matrix.

To determine which gate is selected for a given ROM address, one should consider the low six bits of the address to be a number base four, i.e. digit grouping of two bits. Then the signals ROMAS0-ROMAS3 select the low order digit, ROMAS00-ROMAS30 select the middle digit, and ROMAS000-ROMAS300 sleect the high order digit. The intersection of these signals indicates the gate being selected. For example, to determine which gate is selected for address 41, base 10, one should convert 41 to a base four number. This results in the number 221_4 which indicates that ROMAS200, ROMAS20, and ROMAS1 will be selected. These three signals intersect at the gate whose output is bug 43, pin 8, and feeds connector 58.

In the case that more drive is needed for a given ROM word line, this can be obtained by wiring the corresponding ROMAS address lines to the inputs of one of the four extra NAND gates labelled HA, HB, HC, and HD.

MICROCODE CONTROL CARD

12.1 Objective

The Microcode Control card provides a program counter which supplies the ROM Driver with address information, thus controlling the 64-Word ROM cards. In addition, the card contains two flag flip-flops, two test networks, an adder, and a save register which allows one level of subroutining in the ROM.

12.2 Reference

E & S logic drawing No. 101112-600, Microcode Control.

12.3 Card Contents and Function

Two flip-flops and a group of AND and NOR gates test various conditions indicated by signals from the Directive and Done cards and enabled by ROM signals. The results of these tests form signals T and U and their compliments. These test signals and three 2-bit ROM bytes (YES, NO, SEL) are gated together to determine JUMP or COUNT instructions to the program counter. An adder adds ROM signals ADDR with 1. *EXAD signals from the Binary Numbers card or 2. the contents of the Microcode Control's save register or 3. 0. The resulting eight bits from the input for the program counter (ROM address register). The output of the program counter is decoded and sent to the ROM Driver to select ROM word lines and stored in a save register to be recycled to the adders. Output is also sent to the Clock card for use in the I/O synchronization.

The contents of the program counter are thus conditioned by all of the signals coming into the Microcode Control. Possible operations generated by this input data are: 1. do nothing; 2. add one to the value of the program counter; 3. move the 8-bit ROM address into the program counter; 4. add the ROM address and the eight *EXAD bits from the Small Binary Numbers card to form new contents of the program counter; 5. add the ROM address and the contents of the save register to form new contents for the program counter.

The following discussion will describe the functions of the Microcode Control card in the order in which they were mentioned in the first paragraph of this section.

12.3.1 Test Logic

The test logic is used to generate T and U signals and their compliments. These signals are then used as conditioning signals for instructions to the program counter. The T signals are generated by a 4 x 4 matrix of 4 input AND gates. One input of each gate is tied to Vcc. Another is tied to an input connector signal which indicates some condition of the system. These inputs are labelled TT. ROM signals CI20, CI40, CI100, and CI200 select rows of the matrix, while the ROM signals CI1, CI2, CI14, and CI10 select columns. The intersection of a selected column and a selected row enables the gate that forms that element of the matrix. If care is taken to avoid logical conflicts, more than one condition can be tested simultaneously. Tested signals are logically ORed together to form the signals T and *T.

The U signal is generated by tests of the flag flip-flops and four external conditions (UT). A positive result of any one test drives U high. The flag flip-flops consist of two J-K flip-flops. Clocking is controlled by an external input signal LOADFF. The J side inputs receive data from the Done card (JFL1, JFL2) and the K side inputs are fed by ROM signals (KFL1, KFL2). Both true and compliment outputs are used resulting in both high and low signals, each of which is tied to a 2 input AND gate. These gates are enabled by the ROM signals that select the columns of the T test matrix. The outputs are ORed together.

Four 2-input AND gates are also used to test the UT conditions. The UT signals constitute one input for each of the gates. The ROM signals which select rows of the T test matrix also enable these UT test signals. Again the results are ORed together.

12.3.2 Jump and Count Logic

A long chain of combinatorial logic determines the COUNT and JUMP commands. The YES, NO, and SEL bytes from the ROM, the T and U signals and their compliments, and a signal labelled D from the Done card, which indicates the done condition, are gated together in various combinations to generate the JUMP and COUNT signals.

The truth tables on the logic drawing (sheet 6) show the workings of the circuitry.

These arrays show the conditions under which JUMP or COUNT signals will be generated and which of the test NAND gates is activated.

12.3.3 Adder

Input to hte adder is selected by a two-way switch (2-wide AND-OR-Invert gate) from either the *EXAD bits from the Samll Binary Numbers card or the contents of the save register. The other set of inputs is made up of the ROM signals ADDR. Selection is controlled by the YES bytes as follows:

YES(1)	YES(2)	RESULT
0	X	0 added to ADDR (All 1's are added and Cary In is initiated)
1	0	*EXAD added to ADDR
1	1	Save register contents (*R) added to ADDR

Three 4-bit adders and selection switches are used to make up an 8-bit "conditional sum adder". This arrangement, designed to increase speed, uses two adders to perform the addition of the upper four bits, one assuming a carry and one assuming no carry. The carry out of the lower bit adder selects the appropriate sum. The output of the adders (which is inverted for the upper four bits) forms the input for the program counter.

12.3.4 Program Counter

The program counter is made up of eight SN7471 (or equivalent) J-K flip-flops with AND-OR inputs. The inputs are arranged so that the flip-flops contain compliment information. A clear signal from the Adage control will reset each of the flip-flops so that the program counter contains 0. Clock pulses are sent to each flip-flop. If COUNT is commanded, one is added to the contents of the program counter. If JUMP is commanded, the sum from the adders is registered in the program counter with appropriate adjustments for the upper four bits which are in compliment form. The output of each group of two flip-flops is sent in two bit bytes and their compliments (4 bits in all) to four NAND gates where it is decoded and inverted to form the ROMAS outputs for the ROM Driver and Clock cards.

12.3.5 Save Register

The compliment output of the flip-flops, which is the true value of the program counter since the flip-flops contain compliment information, is also sent to a save register, comprised of D-type latches, for recycling into the adders. The information is clocked into the latches when the ROM signal PUSH R is high, which enables strobing of the latches.

DONE CONDITIONS CARD

13.1 Objective

The purpose of the Done card is to determine when the clipping and dividing operations are finished. As these operations may be terminated under a fairly complex set of circumstances, the logic is involved. The Done card also provides for the "angle detection" function, which is useful in connection with the Warnock hidden line algorithm. Miscellaneous functions are also provided.

13.2 Reference

E & S logic drawing No. 101113-600, Done Conditions Card.

13.3 Card Contents and Function

The block diagram (sheet 1 of the logic drawing) shows the overall layout of the Done card. As shown, three groups of conditions control the done signals. At the bottom of the diagram, the angle detection circuits are represented. These circuits provide signals count, up, and down which control the operations of the angle counters on the Register cards as well as providing signals to the done logic (shown at the top of the diagram) to ensure that clipping will continue at least until the questions about the angle situation are resolved.

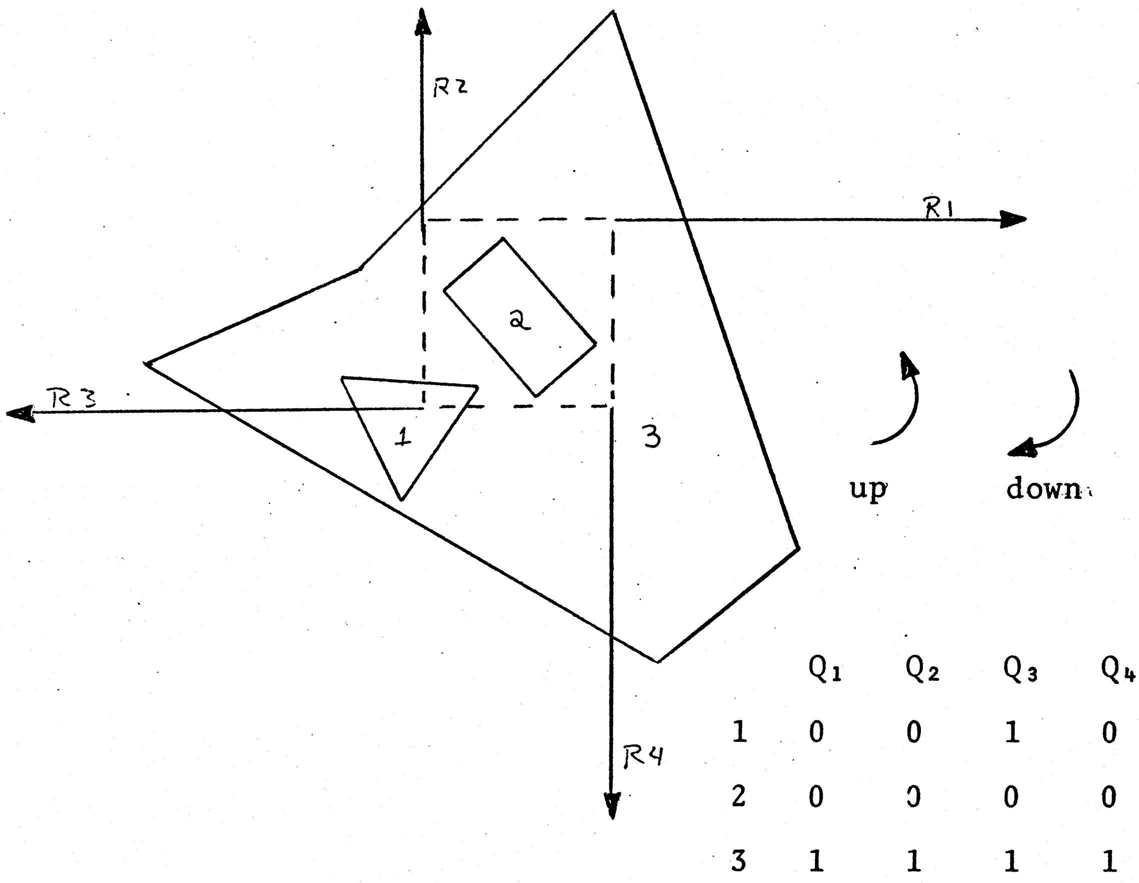
At the upper right of the block diagram are shown the N end clipped and P end clipped logic. These circuits detect when the clipping process has found an adequate solution to the clipping task at hand. When clipping answers have been found for the P (previous) and N (new) ends of the line, the done logic is signaled.

The "divide done" logic is shown at the upper left of the diagram. This logic determines when all four of the division processes are complete and signals the done logic.

The done logic takes the signals developed by the angle and detection and clipping and division termination functions and generates the DONE signal. In addition to the DONE signal, special case sign configuration data from the Sign and Carry card can generate a TRIV REJ signal and special case likeness information can generate CLIP REJ. The following discussions will detail the conditions necessary to develop the DONE signal and then describe the TRIV REJ, CLIP REJ, and miscellaneous functions. Reference should be made to the logic drawing and to figure 13.1.

13.3.1 Angle Detection Logic

The angle detection logic (shown on sheets 5-8 of the logic drawing) was included on the Done card because it is useful in implementing the Warnock hidden line algorithm. The basic purpose of the logic is to detect relationships between polygons and the window of interest. Three cases are significant (see figure below); 1. a polygon which has an edge in the window, 2. a polygon entirely within the window, 3. a polygon which is entirely outside the window. The distinction between the last two cases is most difficult.



	Q1	Q2	Q3	Q4
1	0	0	1	0
2	0	0	0	0
3	1	1	1	1

Counter Contents
(Assuming Counter Clockwise Trace)

Figure 13.1

By summing the angular change referenced from within the window as a point is moved around the polygon in question, it is possible to make the necessary distinction. If the angular change is 360° or some multiple, then the observation point is inside the polygon and the polygon is outside the window. If some corner of the window is within the polygon, it can be detected. And finally, if the angular change is 0°, the observation point is outside the polygon and thus, the polygon lies entirely inside of the window. If polygons are always traced clockwise in reference to the "front", the sign of the angular sum will indicate whether the polygon is seen from the "front" or from the "back".

To implement this angle detection, edges of the polygon are compared to radials emanating at the four corners of the window as shown in Figure 13.1. If the polygon edge (which is also being clipped) intersects the radial in one direction, an "up" count is sent to the counter circuitry of the Register cards. If the line crosses in the other direction, a "down" count is indicated. As the different edges are compared with the radials, the total count for each quadrant (radial) is kept by four separate Register card counters. Perhaps Figure 13.1 explains the function of the angle counters more clearly.

There are four identical angle detection circuits for each of the four quadrants. Appropriate radials are chosen by permuting the inputs to these circuits. A truth table is shown on the logic drawing on each of the four sheets having to do with the angle detection function (sheets 5-8). In each case, the truth table is for the first quadrant but is applicable to any quadrant with appropriate rotations. The entries on the truth table are not immediately meaningful without some decoding and a little mental geometry. DSS means done same side; that is, both ends of the line are on the same side of the radial and no count is needed. DONE means that although the endpoints of the line lie on different sides of the radial, they are situated in such a way that it is impossible for the line to cross.the radial. UP and DOWN indicate that the endpoints lie directly opposite each other on opposite sides of the radial. The little pictures depict the case where the endpoints are diagonally opposite. To understand these little pictures, one should remember that the window is to the left and below. It should also be remembered that any point within the window will stay within the window during the clipping process and that it is the action of the point outside the window that determines counting.

Because the corner of the window, and thus the endpoint of the radial, is treated as between the digital resolution of the Clipping Divider, ambiguity in the angle counts is not possible. Any line which passes directly through the corner of the window will not generate a count.

The logic nets shown on sheets 3-5 implement the angle detection function. A 3-input logical OR collects the conditions which should produce counts. The logic shown on sheet 3 ANDs sign configurations to determine whether to count up or down. The eight input OR gate on sheets 5-8 collects information that indicates that the angle detection process is complete, that is, that no count will be generated or that a count has been generated. The ADONE signal is developed and sent to a flip-flop where it is clocked in and the compliment output sent to a final AND gate controlling the COUNT command. The flip-flop is necessary to ensure that only a single count per line is made.

13.3.2 Clipping Done Logic

The logic that determines when clipping can be terminated for the P (Previous) and N (New) points is shown on sheet 2. Either of two sets of conditions is sufficient to terminate clipping; 1. when data from the Sign and Carry card indicates that the vectors being added by the R & P adders are equal, 2. when a particular point is remembered as being within the window.

The equality data from the Sign and Carry card is inverted and ANDed. The resultant signals are sent to the OR gate in position 66.

The sign configuration of any newly-specified endpoints are clocked into single rank D-type latches when the enabling signal EDGE goes high. The outputs of the latches are sent to logical AND gates whose output feeds the OR gates that control the clipping termination signals. Thus, if an endpoint lies within the window (i.e., it has a sign configuration of +-+-) clipping will immediately be terminated for that endpoint.

If two exactly complimentary numbers are being added in the R & P adders, the midpoint will be on the window edge and clipping can terminate. This condition is indicated by the signals PZRU and NZRU. These signals are direct inputs to the OR gate and are sufficient to terminate clipping.

The outputs of both the N and P side test OR gates are conditioned by more sign configuration data to ensure that the endpoints chosen really are on the screen. If the Clipping

Divider is in CLIP mode, that is if CLIP is high, the clipping process can be terminated for the end in question and the signal *N THRU C or *P THRU C or both may go low. These signals are used by the Poke Control card. If both the P and the N sides have terminated clipping, the DON signal will go high.

13.3.3 Divide Done Logic

Division can be terminated by either of two conditions; 1. the adders of the R & P card are in an all propagate state, or 2. the multiplying adder is being given identical inputs. OR gates test whether one of these conditions is satisfied. When all four units indicate that division is through and the signal DIVIDE is high, the Done signal goes high.

13.3.4 Trivial Rejection Logic

The trivial rejection logic, shown at the upper left of sheet 2, indicates when the line endpoints about to be clipped are both outside the window and on the same edge. When this is true, further clipping with respect to the edge in question is meaningless and the Clipping Divider can move to the next task. 2-input AND gates determine when both endpoints are on one side of a given edge. The resulting signals are ORed to produce TRIV REJ.

13.3.5 Clipping Rejection Logic

The clipping rejection logic is shown next to the trivial rejection logic on sheet 2. It serves basically the same function during the clipping process as the trivial rejection logic does with the initial endpoints. The LIKE signals that form the inputs to this net come from the Sign and Carry card and indicate that the midpoint found was like one or the other end in sign. If the midpoint is on the same side of one edge as one endpoint and on the same side of another edge as the other midpoint, the line can be rejected as it will not intersect the window. The signals are ORed and ANDed together to impliment this function.

13.3.6 Miscellaneous

Sheet four of the done card contains some simple decoding circuits that produce intermediate signals for use in the angle detection logic. Sheet four contains odds and ends.

1. The allow shift logic in the top left corner of the diagram can be used to permit the Ration and Proportion cards to double the information that they would normally use. The allow shift logic is disabled by wiring the allow shift input to ground externally.

2. The CUR logic determines the conditions under which a line must be clipped twice in curve mode. The condition is that one end of the line is in front of the observer and the other end is behind him. The circuit forms the "exclusive or" of two signals which are the sign bits of the Z component of

the coordinates at the time the circuit is used.

3. The J AGREE circuit detects that the new point is remembered to be within the window. This, in turn, will set the "AGREE" flip-flop on the Microcode Control card so that the scope unit can avoid a "setpoint" operation when drawing the next line.

4. The "SET AIC" circuits determine that there is area in common (AIC) between the symbol being considered and the present window. The SET AIC signal goes to the processor to set a testable flag.

5. The "SET HIT" logic sets the hit bit in the processor whenever any part of a line is in the window. The corner count, hit count, and edge count logic increments counters that keep track of how many edges, corners or either are inside the window. The flip-flop at the lower right of the drawing is used to ensure that an edge is counted only once during the clipping process.

SLAVE CONTROL CARD

14.1 Objective

The Slave Control card controls and stores the selection of scopes. The scopes selected by the Channel Control are compared with the CPU permissible scopes. In the case of conflict, a system interrupt signal is developed. The Slave Control card also stores and transmits Z intensity data for use by the Adage Interface card in scope control

14.2 Reference

E & S logic drawing No. 101118, Slave Control Card.

14.3 Card Contents and Function

The Slave Control card contains two single rank registers composed of D-type latches. The register shown at the top of the logic drawing has to do with the selection of scopes. It is fed by input data from the X Register cards. The output of the first eight bits of this register is gated with the output of the second 8 bits to form the scope selection information.

The register shown on the lower part of the logic drawing is the Z intensity register. Input comes from the Y Register cards. The output of this register is sent to one element of 2-way selection switches. The data from the X register cards form the other input for the switches. One of these is selected and becomes the Z intensity data. The following discussion will first describe the circuitry of the scope selection register and the gating controlling its clock pulses and then describe the Z intensity circuitry.

14.3.1 Scope Selection

There are two 8-bit halves to the scope selection register. Input data for both halves comes from the X series of register cards (OUT 2). Both the true and the compliment outputs of each latch are used. The compliment output of the first half of the scope selection register becomes the *SELECT bus which is sent back to the X series register cards. The compliment output of the second half of the register forms the *PERMIT bus which is also sent to the X register cards.

The true output of the first 8 bits is ANDed with the compliment output of the second 8 bits. The resulting signals are ORed together in an expanded OR gate. The signal TUT TUT FORBID goes low if any set of true and compliment latch outputs are both high. TUT TUT FORBID is sent to pin 87 of the Channel Directive cable. The true output of the second half of the register is ANDed with the true output of the first 8 bits. The resulting signals form the *SHOW bus which is sent to the Adage Interface card.

A fairly complex logic controls the enabling of the clock pulse to the scope selection registers. Each half of the register is strobed independently. The strobing of the first half is dependent only on the signal *E BUS COUNT from the

Directive card. The strobing of the second half is also dependent upon *E BUS COUNT but also on an internal signal called PRIVALEDGED. The PRIVALEDGED signal is the compliment output of a D-type latch which is fed by a signal developed by a 2 wide AND-OR-Invert gate. The inputs to this gate are signals from the Matrix Multiplier and Channel Control and the appropriate enabling signal from the Directive card. Which of the signals is enabled is determined by the total system configuration. The clock pulse to the PRIVALEDGED latch is enabled by the ROM signal PDIR.

14.3.2 Z Intensity

The input for the Z intensity register comes from the Y series Register cards (OUT 4). The output of this register goes to one element of a 2-way selection switch (2 wide AND-OR-Invert gate).

Input from the X Register cards (OUT 2) feeds the other element of the output selection switches. The ROM signal READ selects between the two busses. If READ is high, the output of the latches is selected. The *ZINT bus is sent to the Adage Interface and back to the Y series Register cards.

The strobing signal for the entire Z intensity register is dependent on *E BUS COUNT from the Directive and the signal POKE Z, which is tied to the ROM signal READ on the back panel.

14.3.3 LOAD Z

A LOAD Z signal needed by the Adage Interface card is generated by independent circuitry. The basic logic is a single 4-wide AND-OR-Invert gate with two inverters needed to correct the form of the signals.

SMALL BINARY NUMBERS CARD

15.1 Objective

The principle function of the Clipper Small Binary Number card is to setermine offset numbers for updating the ROM program counter upon receiving dispatch codes from the ROM. The card also contains logic to generate the control signals needed for doubling and rounding and a one-shop multivibrator for pulse conditioning the computer's master clear signal.

15.2 Reference

E & S logic drawing No. 101126-600, Small Binary Numbers.

15.3 Card Contents and Function

A three-bit binary dispatch code is sent from the ROM to the Small Binary Number card. The three ROM code bits, called BRADD1, BRADD2, and BRADD3 are sent to a 1 of 10 decoder. BRADD0, the high order bit of the decoder is externally grounded. The output of the decoder is inverted and sent onto 1 Of 8 lines. These lines enable different groups of combinatorial logic which then form a three-bit update number (*EXAD(¢)(7)(8)) sent to the ROM program counter on the Microcode Control Card.

The following discussion will examine the conditions determining the ROM counter update number generated by each of the 8 ROM dispatches and then describe the miscellaneous functions.

15.3.1 Dispatch 0-7

The offset number for dispatch 0, the "done" dispatch, is only dependent on the signal called CUR. The offset number is zero if CUR is high and one if CUR is low. CUR is sent from the Microcode Control card's CUR flip-flop (FF1).

Dispatch 1 is called the "input" dispatch and can generate any of seven ROM update numbers depending on the state of the signals MORE, SELF, DOT, SIZE, SETPT, LINE, BOX, LOAD, ABSOLUTE, and RELATIVE (all Directive card signals). The truth table of Figure 15.1 shows the relationship between the signal states and the corresponding offset numbers. (It is assumed that only one of the signals DOT, SIZE, SETPT, LOAD, and LINE can be high at any one time, and that ABSOLUTE is the compliment of RELATIVE).

Dispatch 2 is called the "more input" dispatch. Its offset number is limited to values ranging from 0-4. Dispatch 2 is dependent on the signals SIZE, DOT, ABSOLUTE, RELATIVE, LINE, SETPT, BOX, and LOAD. Their relationship to the offset numbers is shown in the truth table of Figure 15.2.

In dispatch 3, the "ends" dispatch, any of four offset numbers may be generated. The offset number is dependent on two signals called 3D and CURVE. 3D and CURVE are Directive are Directive signals but are stored in the Adage Control card register. The truth table for dispatch 3 is given in Figure 15.3.

The "load" dispatch is dispatch 4 and has four possible offset numbers (0-3) dependent on five Directive card output signals called LONG, ODD, EVEN, MORE, and WIDE (WIDE is called SELMAT on the Directive card). Figure 15.4 contains the truth table for this dispatch. (According to Directive card logic, only one of the signals ODD, EVEN, and LONG will be high at any one time).

Dispatch 15.5 is called the "setpoint" dispatch and is only dependent on the Directive card signal, SETPT. The offset number is either zero for SETPT high, or one for SETPT low.

Dispatch six is called the "page output to memory" dispatch and is dependent on three signals DOT (a Directive card signal), PTOM and ZTOS (both directive signals stored in the Adage Control card register). The offset number can be any value from 0-3 as shown in the truth table of Figure 15.5.

In dispatch 7, the "divide" dispatch, 4 offset numbers are possible, depending on the states of the signals DOT, STOM, STOS, and AGREE. STOM is a directive stored on the Adage Control card and AGREE is sent from the Microcode Control card's AGREE flip-flop (FF2). The truth table for dispatch 7 is shown in Figure 15.6.

15.3.2 Doubling

Doubling is used to obtain greater accuracy during the division process. The A-side accumulator doubles after each step in the mid-pointing process and thus avoids a premature termination of the division step. To facilitate doubling, the full sum instead of the half sum is poked into the appropriate register and the other register (i.e. the one that was not poked) is allowed to shift. In order to insure that the process will indeed terminate, special injections are added for half of the times that each register is poked.

The selection of the full sum is forced by the ROM signal ALLOW DOUBLING. The same signal forces the half sum to be deselected. These selection signals come in from the small ROM and are sent to the R&P card.

The A-side inject signals developed by the ROM are stored in J-K flip-flops on the Small Binary Numbers card. When the ROM signal DIVIDE is low (indicating that the clipper is not in the divide (DIVI) nor box divide (BDIV) states) the contents of these flip-flops are passed as the injection signals. If DIVIDE is high, the respective carry-out signals (AG(1-24)) are used to control the injection signals thus effecting a 1's compliment arithmetic.

When DIVIDE is low, the B-side injection signals are also simply a copy of the content of the injection flip-flops. If, however, the clipper is in the divide or box divide step, B-side injection is controlled by the poke signals. For each injection signal, two J-K flip-flops store the enabling signals that develop the injection. When the upper register is poked, the contents of the flip-flop corresponding to the upper register is enabled and forms the injection signal. This same enabling signal is wired to both the J and K inputs of the flip-flop so that the flip-flop will be complimented at the next clock pulse. This arrangement results in injecting half the times for every time the upper is selected, and similarly, half the times for every time the lower is selected.

15.3.3 Miscellaneous Functions

The round commands (ROM) are inverted and sent to the Sign and Carry card. A multivibrator is used to condition the computer's "master clear" pulse. OUTX 9 and OUTY 9 from the Register card are inverted and sent to the Adage interface.

OFFSET NUMBER	SIGNAL						
	MORE	SELF	DOT	SIZE	ABSOLUTE	LINE or SETPT	LOAD or BOX
0	1	X	X	X	X	X	X
1	0	1	X	X	X	X	X
2	0	0	0	1	0	X	X
3	0	0	0	1	1	X	X
4	0	0	1	X	X	X	X
5	0	0	0	0	X	1	0
6	0	0	0	0	X	0	1

Figure 15.1: Truth table for dispatch one.

OFFSET NUMBER	SIGNAL				
	DOT	SIZE	ABSOLUTE	LINE or SETPT	BOX or LOAD
0	0	1	0	X	X
1	0	1	1	X	X
2	1	X	X	X	X
3	0	0	X	1	0
4	0	0	X	0	1

Figure 15.2: Truth table for dispatch two.

OFFSET NUMBER	SIGNAL	
	CURVE	3D
0	1	1
1	0	1
2	1	0
3	0	0

Figure 15.3: Truth table for dispatch three.

OFFSET NUMBER	SIGNAL			
	ODD	EVEN	LONG	WIDE or MORE
0	0	1	0	X
1	1	0	0	X
2	0	0	1	1
3	0	0	1	0

Figure 15.4: Truth table for dispatch four.

OFFSET NUMBER	SIGNAL		
	PTOM	DOT	ZTOS
0	0	X	1
1	1	0	X
2	1	1	X
3	0	X	0

Figure 15.5: Truth table for dispatch six.

OFFSET NUMBER	SIGNAL			
	STOM	DOT	STOS	AGREE
0	1	0	X	X
1	1	1	X	X
2	0	X	1	0
3	0	X	1	1

Figure 15.6: Truth table for dispatch seven.

CLIPPER DIRECTIVE CARD

16.1 Objective

The directive card is principally used for storing and decoding the first 16 bits of the clipper operating directive. The directive is sent from either the Matrix Multiplier or the Display Processor, depending on the system configuration.

The card contains 16 D-type latches for storing the directive combinatorial logic for decoding the directive, and the power amplifiers for boosting a special directive signal called SWAP which controls input selection to the clipper register cards. (The use of SWAP is totally dependent on the configuration of the display system and, therefore, its explanation is given in the system description).

16.2 Reference

E & S logic Drawing No. 101116-600, Directive Card.

16.3 Card Contents and Function

One of two 16-bit directives from the selected directive cable is registered into 16 D-type latches when the ROM signal POKE DIRECTIVE goes high. Selection between the two sets of directives is determined by the EINA or EINB signals. If only one set of directives is available (e.g., in a system without a Matrix Multiplier), the A group of directives is disabled and the B group always chosen.

After the directive has been registered, it is decoded by the remainder of the logic on the Directive card. The following discussion will deal primarily with the decoding.

16.3.1 Drawing Command Bits (1-4)

The first four high-order directive bits contain drawing and register transmission information which is used by the Small Binary Number card logic in dispatching to the proper ROM states. Compliment signals are also sent to the Microcode Control. The bits decode according to the truth table shown in Figure 16.1.

16.3.2 Register Address and Load Bits (5-8)

Directive bits 5-8 contain data word length and register selection information for use in register transmission operations. Data word length information is decoded as LONG for data double word length (64-96 bits) and as ODD or EVEN for data single word length (32-48 bits). EVENS corresponds to registers 2 and 4. ODD corresponds to registers 1 and 3.

Register selection information evolves as a 3-bit binary code (denoted as DADD(1), DADD(2), and DADD(3). The three-bit register selection code is gated with a ROM signal called USE DA WRITE. If this signal is high, the three bits are sent directly to the register cards as CODE 1A, CODE 2A, and CODE 3A for the write decoder. And CODE 5A, CODE 6A, and CODE 7A for the read decoder. The combinatorial logic is such that the write inhibit signal (CODE 0A) allows writing if the

ROM signal called WRITE is high and if the binary code is 4 or less (registers 0 through 4). For binary codes larger than 4, writing is only allowed in the count register (E BUS COUNT is enabled).

The read inhibit signal, CODE 4A, is low for binary codes less than 4. For codes greater than 4, reading from registers 0 through 4 is disabled, but reading the count register is allowed (*E COUNT OUT goes low) and the EWIRE gates are enabled (*EWIRE goes low).

If USE DA WRITE is low, then a three-bit binary code from ROM is selected. These bits, denoted as RADD(1), RADD(2), and RADD(3) are subjected to the same register read-write logic as was explained above for the directive binary code.

16.3.3 Data Form Specification Bits (8-12)

The 9th bit of the directive is called SIZE. SIZE going high is a clipper directive which means that input data to the clipper will be in center-size form. This signal is sent to the Small Binary Number card where it is used in determining ROM state dispatches.

The 10th directive bit is called FROM. If FROM is high, the clipper will process new input data as previous point data and the saved data as new point data. If FROM is low, the data processing is reversed. FROM is sent to the Poke Control card for use in accumulator loading control.

Directive bit 11 specifies whether input data is in absolute or relative form. Two Directive card outputs are derived from this bit: the signal called RELATIVE is taken from the zero side of bit eleven's flip-flop and goes high if bit 11 goes high, the signal called ABSOLUTE is taken from the one side. Both signals are sent to the Small Binary Number card. Absolute is also sent to the Poke Control.

Directive bit 12 is called MORE. If MORE is low, the clipper expects one word of data, normally X,Y data. If MORE is high, two words of data are expected: X,Y data in the first word and Z data in the second. Both MORE and *MORE are available outputs. *MORE is sent to the Small Binary Number card and MORE is not used.

16.3.4 Poke Inhibit Bits (13-15)

The next three directive bits, 13, 14, and 15 along with two ROM signals called INHIBM and INHIBS are producing poke inhibit signals for the Poke Control cards. The inhibit signals are called INHIB 1, INHIB 2, INHIB 3, and INHIB 4 on the Directive card and are called INHIBA and INHIBB on the Poke Control cards. If INHIBM and Directive bit 13 are high, INHIB 1 and INHIB 3 are high. If INHIBM is high and directive bit 13 is low, INHIB 2 and INHIB 4 go high. If INHIBS is high and bit 14 is low, INHIB 1 and INHIB 2 go high. If INHIBS is high and bit 15 is low, INHIB 3 and INHIB 4 go high. And finally, if INHIBM and INHIBS go high at the same time, all the inhibit signals go high.

Directive bits 14 and 15 are used in producing three other signals: SELF, INPUT CLEAR X, and INPUT CLEAR Y. If either

DIRECTIVE BITS

1 4	5 8	9	10	11	12	13 15	16
DRAWING COMMANDS	REGISTER ADDRESS AND LOAD TYPE	SIZE	FROM	ABSOLUTE and RELATIVE	MORE	POKE INHIBIT	WIDE

0	0	0	X	LINE
0	0	1	X	SETPOINT
0	1	0	X	BOX
0	1	1	X	DOT
1	X	X	0	LOAD
1	X	X	1	FETCH and MD_1

FIGURE 16.1

bit 14 or 15 go high, SELF goes high. SELF is sent to the Small Binary Number card for use in ROM state dispatch control.

If bit 14 is low and bit 9 (SIZE) is low, and the ROM signal called INPUT STEP is high, INPUT CLEAR X goes high. If bit 9 and bit 15 are low and INPUT STEP is high, INPUT CLEAR Y goes high. INPUT CLEAR X and INPUT CLEAR Y are sent to the Poke Control cards where they are denoted as INPUT and *CS.

16.3.5 WIDE (16)

Bit 16 is used for enabling 72-96 bit loads. The bit only has meaning when a Matrix Multiplier is included in the system.

16.3.6 Directive Card Outputs to the Directive Cable

Two Directive card outputs called MD(1) and MD(2) are sent back to the Channel Control (processor). These bits convey input/output information used by the processor in the register transmission operations. As shown in Figure 16.1, MD(1) is a decode of the first and the fourth directive bits. MD(2) is taken directly from the true side of the bit 13 flip-flop. If MD(1) and MD(2) are both high, the processor uses the contents of its data stack pointer register for a memory address in Clipper register to computer memory, data transfers. In this same mode of data transfer, if MD(1) is high, but MD(2) is low, the Processor uses the contents of its

Read Address register for a memory address. If MD(1) is low and the system is in Clipper data output to memory operation, the processor uses the contents of its Write Address register for a memory address.

CLOCK CARD

17.1 Objective

The Clock card is used to generate and distribute clock pulses within a logic assembly. These clock pulses allow a machine to operate asynchronously from other devices with which it communicates.

17.2 Reference

E & S logic drawing No. 101117-600, Clock Card.

17.3 Card Contents and Function

The Clock card consists of the following logic functions:

1. Clock generation and distribution.
2. Clock PROCEED logic.
3. Control panel gating logic.
4. Input/Output synchronizing logic.

They will be discussed in that order.

17.3.1 Clock Generation and Distribution

The clock is generated by one of three time delay loop arrangements formed by a delay line and a group of Monostable Multivibrators (MM).

This DELAY and MM system has a natural period of oscillation dictated by the value of the delay (T_D) and the time constants of the MM's in the loop.

A control signal named PROCEED (25.11) controls the repetition rate of the Clock in the following sense. If the PROCEED control signal remains high during a clock cycle, or if it comes high at the end of the cycle, the cycle will be repeated at the normal period of oscillations. If, however, PROCEED is low at the end of a clock cycle, the next clock cycle will not be initiated until PROCEED goes high again. The logic for PROCEED is explained in section 17.3.2.

Gate 54 controls opening of the DELAY LOOP, thus enabling or inhibiting the generation of the clock pulses. Figure 17.1 shows a timing diagram for the clock generator. Assume PROCEED has been low for a period of time longer than the natural Clock cycle. As soon as PROCEED goes high, gate 54.8 reinitiates a Clock cycle by going low which, in turn, makes one of the inputs to the delay lines go low (the delay line used is selected by C LONG, C NORMAL, C SHORT from the control panel). After a delay T_D through the delay line, plus delay through gates, 66.8 goes low. When 66.8 goes low, two things happen:

1. Input to gate 64.2 goes low. Gate 64 generates the clock pulse, which is thus controlled by 64.2 or by the delay through the selected delay line. The width of the pulse is thus the delay through the selected line plus an inherent 20 nsec nominal through gates.

2. The MM 65 is triggered, thus initiating a chain of MM's which control the clock rate.

The Mono-stable at 65 has three purposes:

1. To provide a delay between the fall edge of CLOCK and the occurence of LATE CLOCK.
2. To trigger LATE CLOCK at the falling edge of 65.
3. To trigger one of three MM's selected by the position of switch P SHORT, P NORMAL, P LONG in the Control Panel. These MM's select one of three possible clock cycles.

As long as MM 65 or MM 61, 62, or 63 is high, the wide AND gate and 54 is held low which inhibits the repetition of the clock cycle. The microprogram has the ability of increasing the natural period of the clock by bringing the LONG CYCLE signal at C72 high. The long cycle flip-flop is set at the fall of late clock. When long cycle is set, an additional MM (51, 52 or 53 as selected by the control panel switch) is triggered by the signal from the preceeding MM (61, 62, or 63). The additional MM delays repetition of the clock pulse for the entire duration of the pulse produced by the MM, thus elongating the natural frequency of the clock.

17.3.2 Proceed Logic

The proceed logic on the Clock card allows different assemblies within a system to operate asynchronously. The control wire named PROCEED, together with the input/output logic described in paragraph 17.3.4, permits an assembly to stop the clock while waiting for the data transfers to other asynchronous units. Proceed is a function of the input/output waits and of the RUN signal from the control card. For any input channel, the PROCEED signal will be low as long as the ROM has requested an input wait and the transmitting device has not sent a DATA READY signal or the receiving device is still holding its acknowledge signal high. For any output channel, the PROCEED wire will be low as long as the ROM has requested output and the FLAG flip-flop is still high or the receiving device is still holding its acknowledge signal high. PROCEED will also be inhibited by a high signal on *RUN from the control panel.

17.3.3 Control Panel Logic Gating

A particular state of the machine and specified input/output waits can be selected by means of the switches on the control panel. When the device reaches this step or the selected I/O wait is reached, the *RUN line will be driven high or a pulse is produced at the BNC connector (SYNC) on the control panel.

The state of the machine is fed into the Clock card through the ROMAS lines from the Microcode Control card. This state is decoded in binary and sent onto output lines that are sent to the Lamp Driver which drives the state indicating lights on the control panel. The decoded state is compared with the state selected by control panel switches. The comparison is effected through two adders (in positions 16 and 17) which add the data from the control switches and the ones compliment of the data sent from the program counter on the microcode control card (ROMAS). Coincidence is detected by a wide AND gate when the adder output lines contain all 1's.

The coincidence signal is clocked into the flip-flop in position 13 by the LATE CLOCK signal. If coincidence occurs, and if the STOP ON STATE switch on the control panel is on (C69), a STOPPED ON STATE signal is sent to the control panel (Lamp Driver) which in turn drives the *RUN signal high, thus impeding further clocking. The state of flip-flop 13 indicates coincidence of the state of the machine and the state indicated by the control panel switches. This signal is fed regardless of the state of the STOP ON STATE switch to the SYNC hub via the OR logic of gate 11.6.

The MISC TO SYNC signal is ANDed with the signal named DISCRETIONAL. When the MISC TO SYNC signal is high, the DISCRETIONAL signal is available for monitoring at the SYNC hub. If the STOPPED ON MISC switch from the control panel is on, a STOPPED ON MISC signal will be sent back to the Lamp Driver card if the AND conditions are satisfied by MISC TO SYNC and DISCRETIONAL.

17.3.4 Input/Output Synchronization Logic

Figure 17.3.2 shows a timing diagram of the I/O sequences. The clock is equipped to handle 4 input and 4 output channels.

Each output channel consists basically of a J-K flip-flop, an MM for delay purposes, and another J-K flip-flop. The first flip-flop can be considered the flag flip-flop. It is set by a RAISE OUTPUT FLAG signal. When set, it outputs a *FLAG signal which is fed to the clock inhibiting logic and drives PROCEED low. The output also triggers an MM which is used to ensure that data is stabilized on the lines before a DATA READY signal is sent to the device in question. The MM sets the next flip-flop and one input of a NAND gate whose other input is the output of the flip-flop. The signal is inverted once and then a DATA READY signal is sent. The ACK FROM signal that is answered back by the device accepting the input clears both flip-flops which allows clocking to proceed.

The input channel consists of a single J-K flip-flop. When a DATA READY signal is sent from the device transmitting the data, and when the ACK DATA TO signal from the ROM is raised high, an ACK TO signal is sent to the device in question. This signal clears the DATA READY signal from the outputing device, which in turn clears the flip-flop and drives ACK TO low.

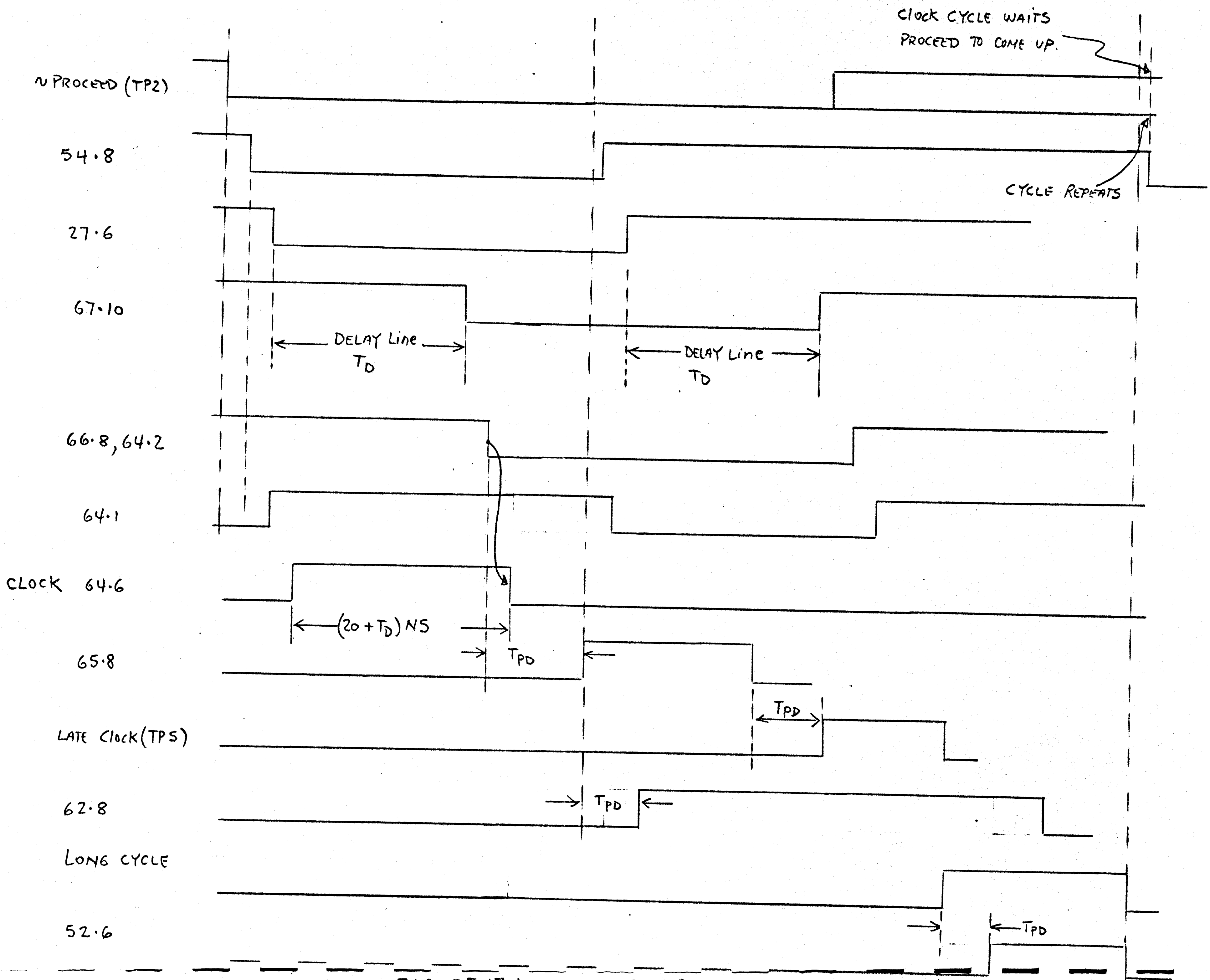
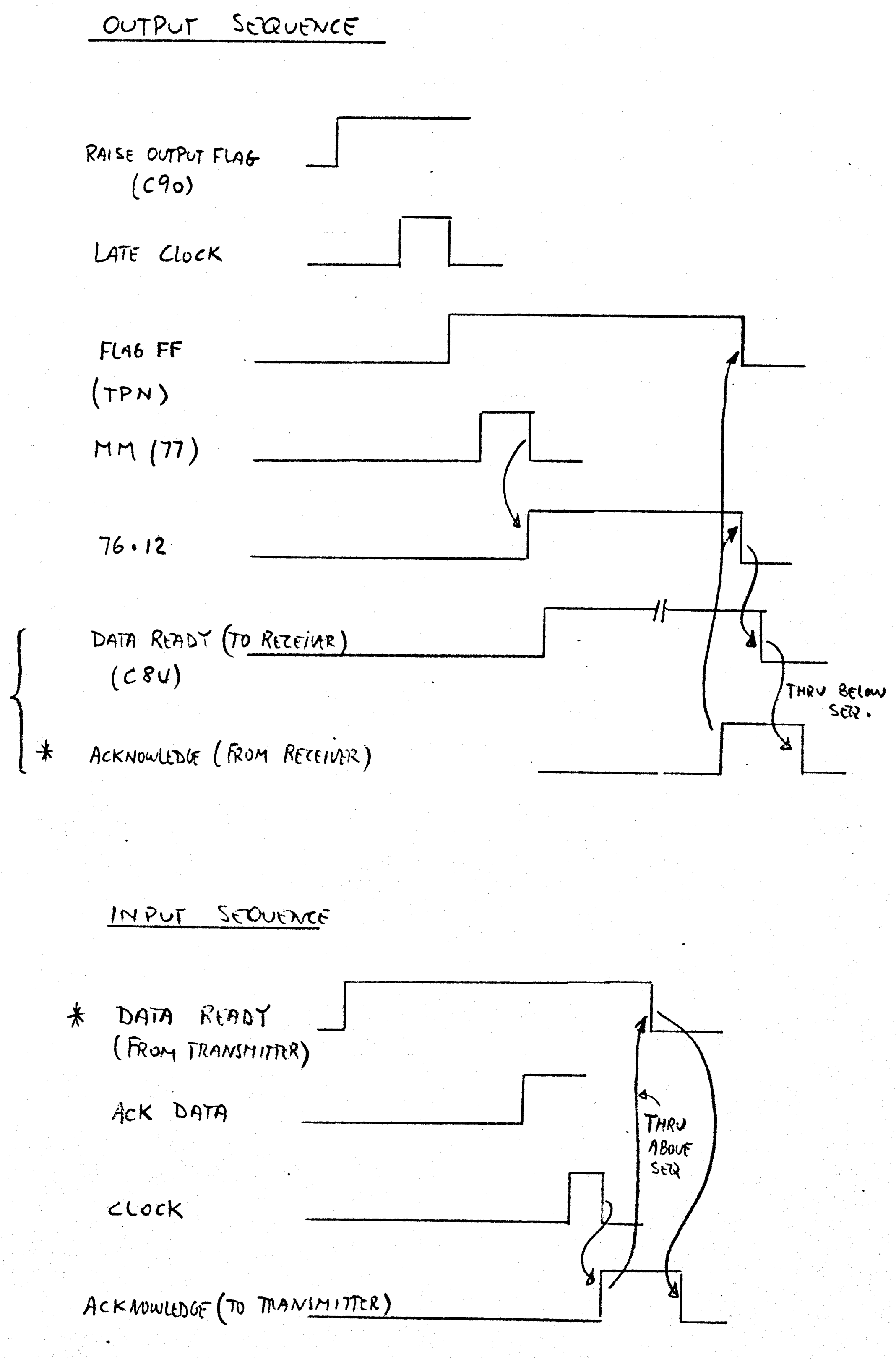


FIGURE 17.1 CLOCK GENERATOR


OUTPUT SEQUENCE

INPUT SEQUENCE

FIGURE 17.2

LAMP DRIVER AND CONTROL PANEL

17A.1 Objective

Two cards constitute the Lamp Driver Assembly. Their purpose is threefold:

1. to drive the control panel lights.
2. to provide interfacing between the control panel switches and other logic in the machine.
3. to allow stopping the clock under certain manual settings in the Control Panel.

17A.2 Reference

E & S logic diagram No. 101115-600, Lamp Driver.

NOTE: The logic diagram for the Lamp Driver includes both boards. IC designations of the form 1-xx refer to board -100; those of the form 2-xx refer to board -101. Connector designations are similar. 1Axx refers to the connectors at the top of board -100, 2Bxx refers to connectors at the bottom of board -101, etc.

E & S logic diagram No. 101117-600, Clock.

17A.3 Card Contents and Function

The Lamp Driver boards are located on the right side of the upper card cage. The board No. 101115-101 is for the most part the Lamp Driver proper. Board No. 101115-100 is the switch interface that interfaces between the switches of the control panel and thelogic of the machine; in particular, the clocking. The two boards will be discussed in the order mentioned.

17A.3.1 Driving the Control Panel Lamps

Board No. 101115-101 is mainly the Lamp Driver proper. All wiring of the Lamp Driver Board is between it and the control panel or the Clock card. All lights are driven by open collector drivers capable of sinking 80 ma at 4 volts. a 200 ohm bleeder resistor to ground allows current to pass through the lamps when the driver is off. This current tends to increase the life of the lamp and provides a slight glow when turned off which allows the determination of whether or not the lamp is still operational even if it is off.

The signals presented to the lights are as follows:

P3 to P8 (white) - Indicates the state of the machine.

PROCEED (green) - Indicates that there is no input-output transaction pending between assembly and other devices. If the clock is stopped, it must be because a manual setting of the switches caused it to stop.

STOPPED (red) - Indicates that a manual setting of the switches has caused the clock to stop. The machine is stopped by one or more of the following conditions: 1. manual clock switch is on, 2. STOPPED ON I/O light is on, or 3. STOPPED ON STATE light is on.

STOPPED ON I/O (white) - Indicates that the clock is stopped because one (or more) of the input-output wait switches is on, the STOP ON I/O switch is on, and the ROM is in I/O wait for the corresponding device. The switch that caused the STOP will have the corresponding light illuminated.

STOPPED ON STATE (white) - Indicates that the clock is stopped because the STOP ON STATE switch is on and the ROM reached the state presented to the switches.

INPUT-OUTPUT WAIT (white) - Indicates conditions as explained in STOPPED ON I/O.

17A.3.2 RUN Flip-Flop

Board No. 101115-100 is the Switch Interface and contains logic that controls the operation of the RUN flip-flop.

Sending of the STOP ON STATE, STOP ON I/O, or single step switches is synchronized by clock pulses to J-K flip-flops (1-41.6; 1-11.6; 1-41.1). This allows one to safely throw any of these switches on when the clock is running without producing faulty triggering of the clock.

17A.3.3 Single Step Clock Pulses

When the SINGLE STEP switch is on (1-41.14 low), the RUN flip-flop (1-11) clears and the *RUN wire at 1BI8 is high. As explained in the Clock card, the *RUN wire causes *PROCEED in the clock to go high, thus stopping the clock. As long as the SINGLE STEP switch is on, the only way to generate clock is by pressing either STEP LEVEL or STEP PULSE. STEP LEVEL (see clock card logic diagram No. 101117-600, sheet 1) at connector 95 by passes the normal clock generator and causes clock to go high as long as the STEP LEVEL switch is on. When the switch is thrown off, the *STEP LEVEL signal triggers MM 65 in order to initiate the late clock chain.

When STEP PULSE is activated, a monostable multivibrator (MM) (1-61) is triggered. During the pulse width of this MM, the RUN flip-flop is set causing *RUN (1BI8) to go low. The time constant in 1-61 was calculated so that the pulse width would not exceed even the shortest natural clock setting. A clock pulse is thus generated via the clock generator. When the pulse in 1-61 ends, the RUN flip-flop goes immediately down again if the SINGLE STEP switch is not thrown OFF.

17A.3.4 STOP ON STATE

The STOP ON STATE switch permits stopping the machine as soon as the ROM reaches the address set in the State switches. When this occurs, the clock card brings *STOPPED ON STATE (2B11) low. This signal is fed to the RUN flip-flop reset (1-11.3), thus motivating *RUN (1BI8) to go high before the next clock cycle is initiated. In order to leave this stopped state, the STEP LEVEL or STEP PULSE switch has to be activated, producing clock as stated before.

17A.3.5 STOP ON I/O

The STOP ON I/O switch enables the INPUT WAIT, OUTPUT WAIT switches for any of several devices to stop the machine whenever a ROM step is reached that calls for an I/O WAIT for a device and the machine is still waiting for I/O from that device(s). Operation of the RUN flip-flop and procedure to leave this STOPPED condition are as described in section 17A.3.4.

17A.3.6 Rotaries

All other switches, except the STATE switches and the clock pulse rotaries are latched to assure proper on/off conditions of the logic signals.

Note that all switches (except the clock pulse rotaries) are connected via a resistor to Vcc. All these switches have their common to ground.

The rotaries, however, have their common to Vcc. The contacts connect Vcc via a 100 ohm limiting resistor to the connector points in this card, where they are connected to ground via a 500 ohm resistor. The voltage at the input to the gate is a nominal 3.5 volts.

The purpose of the CLOCK PULSE WIDTH rotary is to enable one of three possible clock widths in the clock card. The center position is the normal operating position, while the other two decrease or increase the width with respect to the normal.

The PULSE INTERVAL rotary is also a three-position switch which allows increasing or decreasing the clock period with respect to the normal setting.

17A.3.7 SYNC

A BNC hub is provided in the control panel to monitor or SYNC the scope on one of several selected signals within the machine. The signals that drive the BNC hub are as follows:

1. If the CLOCK TO SYNC switch is on, the CLOCK is switched to the BNC connector.
2. If any of the INPUT/OUTPUT WAIR switches are on, the I/O WAIT signal for that device is switched to the BNC.
3. Any time a coincidence occurs between the position of the STATE switches and the state of the machine, the COINCIDENCE signal (see Clock Logic diagram No. 101117-600) is switched to the BNC.
4. When the MISC TO SYNC switch is on and there is

a signal wired (or jumped) to connector (C27) named DISCRETIONAL in the Clock card; this signal is switched to the BNC.

ADAGE INTERFACE

18.1 Objective

The Adage Interface card provides the logical and electrical matching between the clipper and the Adage scope necessary to allow communication.

18.2 Reference

E & S logic drawing No. 101132-600, Adage Interface.

E & S cable assembly drawing No. 101149-100, Adage Cable Assembly.

18.3 Card Contents and Function

The interface to the Adage scope can be considered as an interface to two separate, but interrelated boxes: 1. the Adage digital to analog converter (DAC), and 2. the Adage vector generator. The X, Y, and Z data are fed to the DAC when load lines are enabled as directed by the clipper and allowed by timing synchronization. Data should not be fed to the DAC when the vector generator is sampling data from the DAC. For this reason, a "DAC BUSY" function is generated which inhibits such untimely loading. Slave scope selection data and MOVE/DRAW, DASH LINE directives are sent to the vector generator.

The most important and involved function of the Adage Interface card is in the synchronization and timing required to get the right signals at the right place at the right time.

In the following discussion, the details of DAC loading, directive transfers to the vector generator, the synchronization and timing functions, and finally the registering of directive bits from the Directive cable will be examined.

18.3.1 Loading the DAC

Three control lines control the loading of the DAC. When any one of these lines is pulsed, the DAC will sample and store the contents of the corresponding data lines. These control lines should not be pulsed until after the vector generator has sampled the analog information from the DAC. To prevent untimely loading, the Adage interface card defines a DAC BUSY function which inhibits the three control lines. This function will be described in the section dealing with synchronization and timing (18.3.3).

When a control line is pulsed, data is loaded from the Adage interface card into the DAC. This data is the high 15 bits of X, Y, and Z data developed by the clipper. This data is inverted (except the Z data which is inverted twice bringing it back to true form) and enabled by the *DAC BUSY signal. While *DAC BUSY is low, the data lines contain all 1's.

18.3.2 Directives to the Vector Generator

There are three groups of directive signals to the Adage vector generator. To determine slave scope selection, 8 wires are provided for selection of up to 8 scopes. The MOVE/DRAW directive to the scope (pin 23-22) determines whether the scope does a setpoint (MOVE) or draws a line (DRAW). A DASH LINE directive bit allows the drawing of dashed lines.

All directives are latched in the upper row of D-type latches. Data is clocked in when the *TRANSFER signal goes low. This *TRANSFER signal is conditioned by a logical AND of the DO VG ROM signal and the LOAD signal generated by the multivibrator in position 79. The DASH LINE directive has an additional first line of buffering as it is latched into the D latch 27.2. This latch is strobed by an AND of the clock pulse and PDIR. This signal originates from the directive sent by the clipper. Lines *EINA and *EINB select between one of two possible directives (DIRA(26), DIRB(26)) to form the input to this latch.

In addition to the directives to the vector generator, synchronizing signals are sent and received. A MASTER CLEAR from the clipper clears the Adage vector generator. The TRD signals from the Adage Interface card start the vector generator operations and DAC loading. The vector generator returns a READY TO START signal and a DATA SAMPLES signal. The relationship of these signals in synchronization is explained in the next section.

18.3.3 Synchronization and Timing

The communication between the clipper and the Adage scope is done via a standard "data ready acknowledge" type of synchronization as described in the Clock card writ up (see section 17.4).

For clarity in the ensuing discussion, the signals conditioning synchronization will be listed and defined below:

DO VG A clipper signal indicates that data transfers include directive instructions to the vector generator.

DATA READY TO DISPLAY A clipper signal that indicates that the clipper is ready to display data.

DATA SAMPLED An Adage vector generator signal indicating that data has been sampled from the DAC.

READY TO START An Adage vector generator signal indicating that it is ready to start a display cycle.

DAC BUSY goes high when the LOAD pulse is generated by the MM in position 79 (indicating that the DAC has just been loaded) and stays high until the DATA SAMPLED signal from the Adage DAC clears the flip-flop in position 68.8.

GEN BUSY (vector generator busy) indicates that the vector generator is not through accepting and executing the last commands. GEN BUSY goes high when the MM in 77 pulses the START TRD line and stays high until the READY TO START signal is received from the Adage vector generator.

If the DATA READY TO DISPLAY signal is high and DO VG is low, data is to be loaded into the DAC but not the vector generator. If the DAC is not busy, the AND gate to the monostable multivibrator is satisfied and a *load* signal is produced. This signal enables LOAD X TRD, LOAD Y TRD, and LOAD

Z TRD which are sent to the Adage.

On the trailing edge of the LOAD signal, the flip-flop in position 68.1 is set which generates the ACK FROM DISPLAY signal. If, however, DAC was busy, the AND gate to MM 79 is not satisfied and the sequence waits until DAC BUSY goes low.

If DATA READY TO DISPLAY is high and DO VG is high, directive information is to be sent to the vector generator as well as data loaded into the DAC. The loading sequence proceeds as just described. If GEN BUSY is low, DO VG being high satisfies the MM 78 and a DELAY pulse is generated. This *delay* pulse in turn satisfies MM 77 if GEN BUSY is low. The one shot in 77 produces the START TRD signal which is sent to the Adage vector generator. *Start* TRD starts the display cycle. The vector generator answers the START TRD signal by pulsing DATA SAMPLED which indicates that DAC is now free to accept data, and by bringing READY TO START low. If GEN BUSY is high, the sequence waits for the signal to go low. Thus, while the vector generator is working on one directive, another is waiting at the Adage Interface card.

18.3.4 Directive Register

Bits 18-26 of the directive are sent to the Adage Interface card to be registered in the lower row of D-type latches shown on sheet 3 of the logic diagram. The strobing of information into these latches is conditioned by the AND of the clock pulse and PDIR. 2-way selection switches (2-wide

AND-OR-Invert gates) choose between two sets of directives. *EINA and *EINB enable either the DIRA or DIRB directives respectively. The directive bits envolved are listed below:

STOS
STOM
ZTOS
PTOM, *PTOM
NTOM
3D
CURVE
MEF

APPENDIX

The appendix includes documents that define or describe the operation of the Clipping Divider. The following documents are contained:

19.1 Algorithm
The definition of the algorithm in terms of PDP-9 macros. The ROM bits activated by each step are listed in octal codes that relate to the ROM bit descriptions that preceed the algorithm.

19.2 ROM bit wiring
Lists the ROM bits and their wiring.

19.3 Registers and busses of the Clipping Divider
Shows the registers and busses of the clipper in block diagram form.

19.4 ROM 8X16 bit descriptions
Shows the contents of the 8X16 ROM and the small ROMs on the Poke Control card. This bit pattern controls poking and input andoutput switches and is used to decode the S group of algorithm ROM bits.

19.5 ROM 64 Word bit pattern
Shows the content of the 6 ROM cards used in the clipper.

19.6 ROM 64 Word component placement
Defines the component placement in the ROM cards.

```
        .TITLE CLIPAL

/ROUNDC         6 BITS OF ROUNDING CODE
        /AUTOROUND ODD  /VALID THIS PULSE
        /INJECT P ODD   /VALID NEXT PULSE
        /INJECT N ODD   /VALID NEXT PULSE
        /AUTOROUND EVEN /VALID THIS PULSE
        /INJECT P EVEN  /VALID NEXT PULSE
        /INJECT N EVEN  /VALID NEXT PULSE
/ENABLC         4 BITS TO CONTROL "OUT CODE" IN SANDC
        /ENABLE ODD +
        /ENABLE ODD -
        /ENABLE EVEN +
        /ENABLE EVEN -
/STOREC         8 BITS TO CONTROL REGISTER STORAGE
        /USE DIRECTIVE ADDRESS FOR READ
        /USE DIRECTIVE ADDRESS FOR WRITE
        /WRITE
        /ECOMP  /REGISTER COMPLEMENT OUTPUT TO ACCUMULATORS
        /ETRUE  /REGISTER TRUE OUTPUT TO ACCUMULATORS
        /3 BIT REGISTER ADDRESS
/LROMC 12 BITS, 3 FOR EACH OF 4 LITTLE ROM ADDRESSES FOR
        /UPPER SWITCH
        /SUM SWITCH
        /LOWER SWITCH
        /OUTPUT SWITCH
/STEPC 9 BITS TO MARK SPECIFIC STEPS
        /INPUT 1        /WAITING FOR INPUT AND DIRECTIVE
        /INPUT 2        /WAITING FOR SECOND DATA, DSR SAYS WHICH HALF
        /CLIP-TYPE SHIFT ALLOWED
        /DIVIDE-TYPE SHIFT ALLOWED
        /INPUT STEP, SPECIAL CLEARING ALLOWED BY X AND Y
        /BOXEDGE        /MARKS EDGE STEP IN BOXING MODE
        /CLIP
        /DIVIDE
        /EDGE   /MARKS EDGE STEP IN ANY MODE INCLUDING BOXING
/FLOPC 8 BITS OF FLIP-FLOP CONTROL
        /FORCE HIT
        /ALLOW SETPOINT HIT
        /ALLOW AIC TO SET
        /CLEAR AIC
        /ALLOW AGREE TO SET
        /CLEAR AGREE
        /ALLOW CUR TO SET
        /CLEAR CUR
/INOUTC         13 BITS OF INPUT AND OUTPUT FLAG CONTROL
        /DO VG
        /ADAGE TAKE Z ONLY
        /ADAGE TAKE X AND Y (AND Z IF NO DEPTH)
        /ADAGE TAKE X Y AND Z
        /POKE DIRECTIVE
        /INPUT WAIT
        /ACKNOWLEDGE INPUT FLAG
        /RAISE OUTFLAG FOR FETCHED DATA
        /OUTPUT WAIT
        /RAISE OUTFLAG FOR MEMORY DATA
        /OUTPUT WAIT
        /RAISE OUTFLAG FOR SCOPE
        /OUTPUT WAIT
```

```
/POKEC 13 BITS OF POKE CONTROL DATA
       /POKE Z & READ
       /INHIBIT EXCEPT SELF DATA
       /INHIBIT EXCEPT FOR MORE DATA
       /CLEAR PU,PL,NU,NL      /4 BITS
       /ONLY ON, ONLY OFF, ONLY REV    /3 BITS FOR BOXING CASES
       /POKE ROM-ADDRESS       /3 BITS
/JUMPA 8 BITS OF JUMP ADDRESS
/JUMPC 18 BITS OF JUMP CONTROL
       /3 BITS OF SMALL NUMBER SELECTION
       /PUSH FOR SUBROUTINE ENTRY
       /2 YES BITS
       /2 NO BITS
       /2 SEL BITS
       /8 TEST BITS
/TEST CONDITIONS
/2ØØ   REJECT  /LOAD   FETCH   ZP-     ZN-
/1ØØ   TRIV REJ        /BOX    A TO M  P TO M  Z TO M
/4Ø    SPARE   /72 REG EVEN ADDR       S TO M  S TO S
/2Ø    SPARE   /3D     LINE    *P TO M DOT
/U             /J AGREE        K AGREE J CUR   K CUR
       /U      /13     /4      /2      /1

       .DEFIN CODES,R,E,S,L,T,F,I
       R
       E
       S
       L
       T
       F
       I
       .ENDM
       .DEFIN POKEC,I,C,B,P
       I*2Ø+C*1Ø+B*1Ø+P
       .ENDM
       .DEFIN JUMPA,A  /JUMP ADDRESS
       A-BROMT-2/15
       .DSA A
       .ENDM
       .DEFIN JUMPN,A
       A
       A
       .ENDM
       .DEFIN JUMPC,D,Y,N,S,B
       D*1Ø+Y*4+N*4+S*4ØØ+B
       .ENDM
       .DEFIN JUMP,A   /UNCONDITIONAL JUMP
       JUMPA A
       JUMPC Ø,1,Ø,1,Ø
       .ENDM
       .DEFIN DISP,A,D /UNCONDITIONAL DISPATCH
       JUMPA A
       JUMPC D,2,Ø,1,Ø
       .ENDM
       .DEFIN JUMPI,A,B        /JUMP IF CONDITION
       JUMPA A
       JUMPC Ø,1,Ø,Ø,B
       .ENDM
       .DEFIN JUMPU,A,B        /JUMP UNLESS CONDITION
       JUMPA A
```

>

```
        JUMPC Ø,1,Ø,1,B
        .ENDM
        .DEFIN DISPI,A,D,B      /DISPATCH IF CONDITION
        JUMPA A
        JUMPC D,2,Ø,Ø,B
        .ENDM
        .DEFIN DISPU,A,D,B      /DISPATCH UNLESS CONDITION
        JUMPA A
        JUMPC D,2,Ø,1,B
        .ENDM
        .DEFIN NEXT     /GO ON TO NEXT INSTRUCTION
        JUMPN Ø
        JUMPC Ø,Ø,Ø,Ø,Ø
        .ENDM
        .DEFIN DCALI,A,D,B      /DISPATCH SUBROUTINE CALL
        JUMPA A
        JUMPC D,6,Ø,Ø,B
        .ENDM
        .DEFIN EXIT
        JUMPN 1
        JUMPC Ø,3,Ø,1,Ø
        .ENDM
        .DEFIN EXITU,A
        JUMPN 1
        JUMPC Ø,3,Ø,1,A
        .ENDM
        .DEFIN POKEA    /POKE ALL
        POKEC Ø,Ø,Ø,Ø
        .ENDM
        .DEFIN CLEAR,W  /CLEAR SELECTED
        POKEC Ø,W,Ø,Ø
        .ENDM
        .DEFIN POKE,N   /POKE SELECTED
        POKEC Ø,Ø,Ø,N
        .ENDM
        .DEFIN CLEPK,C,P        /CLEAR AND POKE
        POKEC Ø,C,Ø,P
        .ENDM
        .DEFIN APOKE    /ARBITRARY POKE
        POKEC Ø,Ø,Ø,Ø
        .ENDM
        .DEFIN NPOKE    /NO POKE
        POKEC 3,Ø,Ø,Ø
        .ENDM
        .DEFIN BPOKE A,B
        POKEC Ø,Ø,B,A
        .ENDM
        .DEFIN PATH,S,L
        CODES Ø,Ø,S,L,Ø,Ø,Ø
        .ENDM
        .DEFIN OUTPUT,L,I
        CODES Ø,Ø,Ø,L,Ø,Ø,I
        .ENDM
        .GLOBL BROMT
BROMT   BROMTL-BROMT-2/15
        15

CLEA    .ASCII .CLEA.
        CODES Ø,Ø,Ø,ØØØØ,Ø,25,Ø /CLEAR AIC AGREE CUR
        APOKE
>
```

```
        JUMP WAIT

 /ORIGIN FOR DISPATCH Ø

 DONE=.
 CFET   .ASCII .CFET.
        PATH 1Ø,2Ø2Ø     /SAVE TO N
        POKE Ø  /POKE N
        JUMPA CFEU
        JUMPC Ø,1,Ø,2,2 /JUMP IF CUR

 WAIT   .ASCII .WAIT.
        CODES Ø,Ø,Ø,ØØØØ,4ØØ,1,6ØØ
        POKEA
        NEXT

 DATA   .ASCII .DATA.
        CODES Ø,Ø,1Ø,1Ø2Ø,2Ø,Ø,1ØØ
        POKEA
        DISPU MORE,1,2Ø4         /UNLESS FETCH

 FETC   .ASCII .FETC.
        PATH 21Ø,2Ø2Ø
        POKEA
        JUMPU FETR,54   /UNLESS ODD OR 72 BIT

 FETL   .ASCII .FETL.
        OUTPUT Ø2Ø1,14
        POKE 2  /UPPERS ARE INTERCHANGED
        JUMPU WAIT,5Ø   /UNLESS 72 BIT REG

 FETR   .ASCII .FETR.
        OUTPUT Ø1Ø1,14
        APOKE
        JUMP WAIT

 /ORIGIN FOR DISPATCH 1

 MORE   .ASCII .MORE.
        CODES Ø,Ø,1Ø,1Ø2Ø,22Ø,Ø,3ØØ     /SAVE TO LOWER, DATA TO UPPER
        POKEC 1,Ø,Ø,Ø
        DISP SIZR,2

 SELF   .ASCII .SELF.
        PATH 13,2ØØØ
        POKEC 2,1Ø,Ø,2
        DISP SIZR,2

 /ORIGIN FOR DISPATCH 2

 SIZR   .ASCII .SIZR.
        CODES 22,Ø,1Ø,4Ø2Ø,Ø,Ø,Ø
        POKE 5
        JUMP ENDN

 SIZA   .ASCII .SIZA.
        CODES 22,Ø,Ø,4ØØØ,Ø,Ø,Ø
        CLEPK 5,5
        JUMP ENDN

>
```

```
DOTT    .ASCII .DOTT.
        PATH Ø,Ø31Ø
        CLEPK 1Ø,5
        NEXT

ENDR    .ASCII .ENDR.     /END STEP SET SAVE
        PATH 4Ø,Ø3Ø2      /NEW END TO SAVE, ENDS TO AC'S
        POKEA
        DISP CURV,3

ENDN    .ASCII .ENDN.     /END STEP PRESERVE SAVE
        PATH Ø,Ø3ØØ
        POKEA
        DISPU CURV,3,31Ø          /UNLESS LOAD OR BOX
/END OF DISPATCHES 1 AND 2.

LOAD    .ASCII .LOAD.
        CODES Ø,Ø,13,316Ø,Ø,4,Ø /INST TO D,MASTER SWAP TO D
        POKEA   /CLEAR AGREE
        DISPU LOAE,4,11Ø          /UNLESS BOX

BOXR    .ASCII .BOXR.
        PATH 43,Ø6Ø1      /WRITE MASTER AS BOX IN INST REG
        NPOKE
        JUMP BWIN

/ORIGIN FOR DISPATCH 4

LOAE    .ASCII .LOAE.
        PATH 35Ø,Ø6Ø6     /LOAD EVEN ADDRESS (L,B)
        APOKE
        JUMP WAIT

LOAO    .ASCII .LOAO.
        PATH 35Ø,Ø6Ø4     /LOAD ODD ADDRESS (R,T)
        APOKE
        JUMP WAIT

LOAA    .ASCII .LOAA.
        PATH 14Ø,Ø6Ø2
        APOKE
        JUMP WAIT

LOAR    .ASCII .LOAR.     /LOAD AS A RECTANGLE
        CODES 33,Ø,162,Ø621,Ø,2Ø,Ø        /NEW TO U; -WIN TO L
        POKEA   /CLEAR AIC
        NEXT

/END OF DISPATCH 4

TAIC    .ASCII .TAIC.     /TEST AREA IN COMMON
        CODES Ø,Ø,Ø,ØØØØ,1Ø,4Ø,Ø          /ALLOW AIC TO SET
        APOKE
        JUMP WAIT

/ORIGIN OF DISPATCH 3

CURV    .ASCII .CURV.
        CODES Ø,Ø,43,ØØØ3,Ø,2,Ø /WRITE INST, ALLOW CUR TO SET
        APOKE
>
```

```
        JUMPI COMP,2Ø2  /JUMP IF ZP-

WIN3    .ASCII .WIN3.
        PATH Ø,Ø53Ø     /NO INJECT, X=+Z OR X=-Z IS ON
        POKE 1
        DISP SETP,5

SPA1    .ASCII .SPA1.   /SPARE
        PATH Ø,ØØØØ
        APOKE
        JUMP WAIT

WIN2    .ASCII .WIN2.
        CODES 3Ø,Ø,22,ØØ2Ø,Ø,Ø,Ø
        POKE 6  /LOWERS
        DISP SETP,5

/END OF DISPATCH 3
/CURVE MODE SPECIAL STEPS

CFEU    .ASCII .CFEU.
        CODES Ø,Ø,13,2ØØØ,Ø,1,Ø /INST TO P,CLEAR CUR
        POKE 5
        JUMPU WIN3,2Ø1  /UNLESS ZN-

COMP    .ASCII .COMP.
        CODES 33,Ø,Ø,ØØ4Ø,Ø,Ø,Ø
        CLEAR 12        /CLEAR UPPERS, POKE LOWERS
        NEXT

COMQ    .ASCII .COMQ.
        PATH Ø,33ØØ
        POKEA
        JUMP WIN3

/ORIGIN FOR DISPATCH 5

SETP    .ASCII .SETP.
        CODES Ø,Ø,Ø,ØØØØ,Ø,1Ø5,Ø        /ALLOW SETPOINT HIT
        APOKE   /CLEAR CUR AND AGREE
        DISP CFET,Ø

EDGE    .ASCII .EDGE.
        CODES Ø,Ø,Ø,Ø3ØØ,1,ØØ,Ø /ALLOW CORNER HIT COUNT
        POKEA
        NEXT

/END OF DISPATCH 5

CLIP    .ASCII .CLIP.
        CODES 44,6,Ø,ØØØØ,4,Ø,Ø
        POKEA
        JUMPA DONE
        JUMPC Ø,2,2,2,2ØØ       /DISPATCH IF REJ

TANC    .ASCII .TANC.
        CODES 3Ø,Ø,Ø,Ø53Ø,Ø,Ø,Ø
        POKE 1
        DISPI ZTOS,6,32 /IF 3D OR NO OUTPUT

>
```

```
COUT    .ASCII .COUT.
        CODES Ø3,Ø,12,Ø12Ø,Ø,Ø,Ø
        POKE 4  /WR AND X-WR TO EVENS
        NEXT

COUU    .ASCII .COUU.
        PATH Ø,Ø3ØØ       /X TO EVENS
        POKE 4
        JUMPI NPTM,21     /IF DOT

PPTM    .ASCII .PPTM.
        CODES 1Ø,Ø,Ø,Ø4Ø1,Ø,Ø,14
        POKE 4
        JUMPI NXTM,3Ø     /IF 3D

NPTM    .ASCII .NPTM.
        OUTPUT Ø4Ø2,14
        POKE 4
        JUMPU VIEW,1Ø1   /UNLESS Z TO S

/ORIGIN FOR DISPATCH 6

ZTOS    .ASCII .ZTOS.
        CODES 3Ø,Ø,Ø,ØØØ5,Ø,Ø,4ØØ3        /WRITE IN Z REGISTER
        POKEC 4,Ø,Ø,4
        JUMP VIEW

PXTM    .ASCII .PXTM.
        OUTPUT ØØØ1,14
        POKE 4
        JUMP PPTM

NXTM    .ASCII .NXTM.
        OUTPUT ØØØ2,14
        POKE 4
        JUMP NPTM

VIEW    .ASCII .VIEW.
        CODES Ø,Ø,11,2Ø2Ø,Ø,2ØØ,Ø         /FORCE HIT
        POKE 4  /VIEWPORT TO EVENS
        JUMPU DIVI,1Ø4  /UNLESS A TO M

/END OF DISPATCH 6

ADTM    .ASCII .ADTM.
        CODES Ø,Ø,14,ØØØ4,Ø,Ø,14          /NAME TO OUTPUT
        NPOKE
        NEXT

DIVI    .ASCII .DIVI.
        CODES Ø,14,Ø,ØØØØ,42,Ø,Ø
        POKEA
        JUMPA DONE
        JUMPC Ø,2,2,1,43          /DISPATCH UNLESS SCALED OUTPUT

DDIS    .ASCII .DDIS.
        CODES Ø,Ø,Ø,ØØØØ,Ø,4,Ø  /CLEAR AGREE
        APOKE
        DISP PSTM,7

>
```

```
/ORIGIN FOR DISPATCH 7

PSTM    .ASCII .PSTM.
        OUTPUT Ø4Ø1,14
        APOKE
        NEXT

NSTM    .ASCII .NSTM.
        OUTPUT Ø4Ø2,14
        APOKE
        DISPU CFET,Ø,41 /UNLESS SCALED TO SCOPE

PSTS    .ASCII .PSTS.
        OUTPUT Ø4Ø1,12ØØ3          /TAKE X Y (Z) TO SCOPE
        APOKE
        NEXT

NSTS    .ASCII .NSTS.
        CODES Ø,Ø,Ø,Ø4Ø2,Ø,1Ø,11ØØ3      /ALLOW AGREE TO SET
        APOKE
        DISP CFET,Ø

/THE BOXING ROUTINE

BWIN    .ASCII .BWIN.
        CODES 3Ø,Ø,22,Ø72Ø,Ø,Ø,Ø         /INST TO UP; -WIN TO LO
        POKEA
        NEXT

BEDG    .ASCII .BEDG.
        CODES Ø,Ø,Ø,Ø3ØØ,11,Ø,Ø
        POKEA
        JUMPA WAIT
        JUMPC Ø,1,Ø,2,1ØØ          /JUMP IF TRIV REJ

BTRN    .ASCII .BTRN.   /BOX TRANSPOSE
        PATH Ø,Ø72Ø
        BPOKE 7,2
        NEXT

BRTU    .ASCII .BRTU.
        PATH Ø,Ø1ØØ
        BPOKE 2,1
        NEXT

BSWU    .ASCII .BSWU.   /SWAP UPPERS
        PATH Ø,Ø2ØØ
        BPOKE 2,1
        NEXT

BMAS    .ASCII .BMAS.
        PATH 13,225Ø
        POKE 1
        NEXT

BVPT    .ASCII .BVPT.
        PATH 11,225Ø
        BPOKE 4,4
        NEXT
```

>

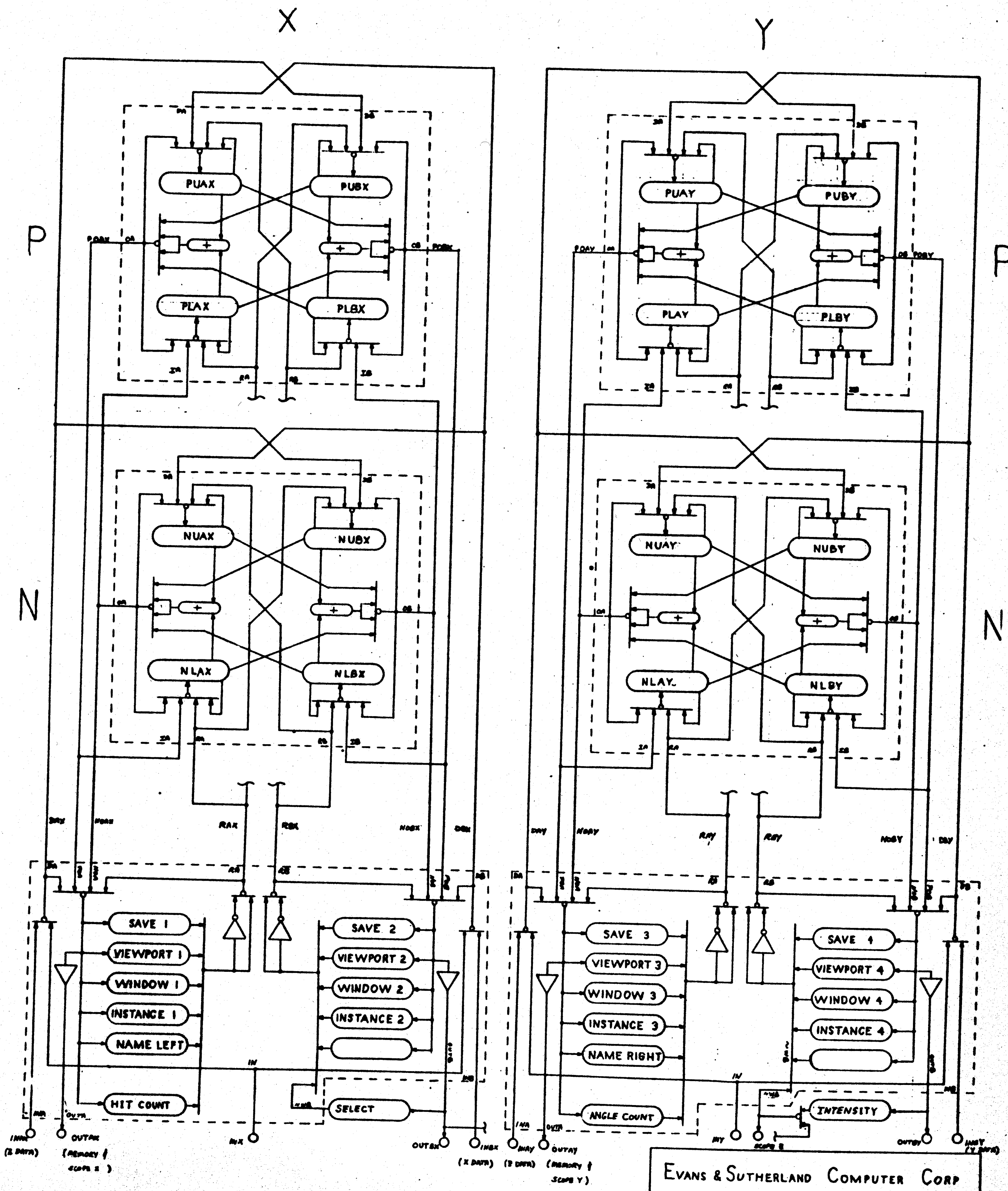
```
BDIV    .ASCII .BDIV.
        CODES Ø,14,Ø,ØØØØ,42,Ø,Ø
        POKEA
        JUMPN Ø
        JUMPC Ø,Ø;2,Ø,Ø /REPEAT TILL DONE

BOLM    .ASCII .BOLM.
        PATH 13,24ØØ    /COMPUTED ANSWERS TO LOWERS AND
        POKEA   /SWAPPED OLD MASTER TO UPPERS
        NEXT

BRPM    .ASCII .BRPM.   /REPLACE MASTER WITH COMPUTED VALUE
        PATH Ø,Ø1ØØ
        BPOKE 2,2       /ONLY IF OFF
        NEXT

BWNW    .ASCII .BWNW.   /WRITE NEW WINDOW
        PATH 42,Ø2Ø1
        POKE 2
        NEXT

BOLV    .ASCII .BOLV.
        PATH 11,2ØØØ
        POKE 2  /VIEWPORT TO UPPERS
        NEXT

BSWV    .ASCII .BSWV.   /SWAP VIEWPORT
        PATH Ø,Ø2ØØ     /SWAP UPPERS
        BPOKE 2,1       /IF OFF BACKWARDS
        NEXT

BRPV    .ASCII .BRPV.   /REPLACE VIEWPORT
        PATH Ø,Ø1ØØ
        BPOKE Ø,4       /ONLY IF ON
        NEXT

BWNV    .ASCII .BWNV.   /WRITE NEW VIEWPORT
        PATH 41,Ø2Ø1
        APOKE
        JUMP WAIT

BROMTL Ø
        .END
END OF FILE REACHED BY:
P 6Ø
>
```

ROM NAME	WIRE NAME	ROM CONNECTION	CONNECTION		REMARKS
RND1	AR	26.87	38.18		
RND2	AIP	26.82	38.33		
RND3	AIN	26.85	38.35		
RND4	BR	26.9Ø	38.2Ø		
RND5	BIP	26.84	38.37		
RND6	BIN	26.86	38.39		
ENA1	EA+	26.88	4.94		
ENA2	EA-	26.83	4.71		
ENA3	EB+	26.89	4.91		
ENA4	EB-		4.5Ø	4.2	NOT USED IN ROM
STR1	USEDA READ	26.14	44.9		
STR2	USEDA WRITE	26.13	44.7		
STR3	WRITE	26.12	44.7Ø		
STR4	*ECOMP	26.11	17.34		
STR5	*ETRU	26.81	17.21		
STR6	RADD(1)	26.15	44.3		
STR7	RADD(2)	26.6	44.5		
STR8	RADD(3)	26.1Ø	44.19		
LRO1	ALPHAD2	27.8	8.8		
LRO2	ALPHAD1	27.81	8.7		
LRO3	ALPHADØ	27.16	8.6		
LRO4	BETAD2	27.1Ø	8.28		
LRO5	BETAD1	27.88	8.27		
LRO6	BETADØ	27.14	8.26		
LRO7	LE(1)	27.82	9.87		
LRO8	LE(2)	31.14	9.88		
LRO9	LE(3)	27.12	9.85		
LR1Ø	GAMMAD2	27.11	8.78		
LR11	GAMMAD1	27.87	8.77		
LR12	GAMMADØ	31.82	8.76		
STE1	CLIPPER SETTLED	26.8	49.69		
STE2		26.9			NOT USED IN LOGIC
STE5	INPUT STEP	26.7	44.3Ø		
STE6	BOXING	26.6	9.66		
STE7	CLIP	27.15	36.76		
STE8	DIVIDE	27.7	36.45		
STE9	EDGE	27.9	36.75		
FLO1	FORCE HIT	28.9Ø	36.5		
FLO2	ALLOW HIT	28.88	36.6		
FLO3	ALLOW AIC	28.82	36.9		
FLO4	CLRAIC	28.8	49.75		
FLO5	ALLOW AGREE	28.7	36.22		
FLO6	KFL1	28.84	34.5Ø		
FLO7	ALLOW CUR	28.15	36.24		
FLO8	KFL2	28.85	34.46		
INOØ	DOVG	31.16	46.96		

>

```
  IN01    TAKEZ    28.13    42.69
  IN02    LOADXY   28.12    46.93
  IN03    LOADXYZ  28.16    46.95
  IN04    POKE     28.1Ø    42.11
          DIRECTIVE
  IN05    INPUT    31.8     4Ø.47
          WAIT
  IN06    ACK      31.7     4Ø.7Ø
          INPUT
  IN09    OUTFLAG  31.15    4Ø.9Ø
          MEM
  IN1Ø    OUTPUT   31.15    4Ø.65
          WAIT MEM
  IN11    OUTFLAG  31.9     4Ø.92
          SCOPE
  IN12    OUTWAIT  31.9     4Ø.63
          SCOPE
  PKCØ    POKEZ    28.89    42.8Ø    42.68
  PKC1    INHIBS   3Ø.88    44.29
  PKC2    INHIBM   3Ø.84    44.27
  PKC3    CLEARPU  29.83     9.23
  PKC4    CLEARPL  29.87     9.27
  PKC5    CLEARNU  29.9Ø     9.25
  PKC6    CLEARNL  29.81     9.26
  PKC7    ONLYON   29.89     9.53
  PKC8    ONLYOFF  29.82     9.51
  PKC9    ONLYREV  3Ø.85     9.54
  PK1Ø    PE(1)    29.85     9.44
  PK11    PE(2)    29.84     9.43
  PK12    PE(3)    29.86     9.42
  JUA3    ADDR(3)  28.87    34.24
  JUA4    ADDR(4)  28.86    34.23
  JUA5    ADDR(5)  28.14    34.82
  JUA6    ADDR(6)  28.81    34.81
  JUA7    ADDR(7)  31.88    34.8Ø
  JUA8    ADDR(8)  28.11    34.65
  JUC1    BRADD1   3Ø.81    38.22
  JUC2    BRADD2   29.16    38.11
  JUC3    BRADD3   29.14    38.1Ø
  JUC4    PUSHR             34.7     34.2      NOT USED IN ROM
  JUC5    YES(1)   29.11    34.6
  JUC6    YES(2)   3Ø.82    34.5
  JUC7    NO(1)    3Ø.11    34.38
  JUC8    NO(2)             34.37    34.2      NOT USED IN ROM
  JUC9    SEL(1)   3Ø.15    34.39
  JU1Ø    SEL(2)   3Ø.14    34.4Ø
  JU11    CI2ØØ    29.15    34.56
  JU12    CI1ØØ    29.9     34.29
  JU13    CI4Ø     3Ø.9     34.16
  JU14    CI2Ø     3Ø.87    34.43
  JU15    CI1Ø     29.88    34.52
  JU16    CI4      3Ø.1Ø    34.49
  JU17    CI2      3Ø.16    34.48
  JU18    CI1      31.11    34.45
END OF FILE REACHED BY:
P 6Ø
>
```

19.3



EVANS & SUTHERLAND COMPUTER CORP

SALT LAKE CITY, UTAH

CLIPPING DIVIDER REGISTERS AND BUSSES	SHEET 1 OF 1 DATE 1-24-1969

101123-900

SWITCH CONTROL

```
        .TITLE LROMT
        .DEFINE LROM P1,P2,N1,N2
        P2*2Ø+P1*2Ø+N2*2Ø+N1
        .ENDM
        .DEFIN PLROM,PU,PL,NU,NL
        NL*2Ø+PL*2Ø+NU*2Ø+PU
        .ENDM
        .GLOBL PLROMT,SLROMT
/TABLE FOR POKES
PLROMT=.
        PLROM 17,17,17,17       /Ø = POKE ALL
        PLROM 12,17,12,17       /1 = POKE TRANSPOSE & COMP
        PLROM 17,Ø,17,Ø /2 = POKE UPPERS
        PLROM Ø,Ø,17,17 /3 = POKE N
        PLROM 12,12,12,12       /4 = POKE EVEN
        PLROM 17,17,Ø,Ø /5 = POKE P
        PLROM Ø,17,Ø,17 /6 = POKE LOWERS
        PLROM 12,5,5,12 /7 = BOX TRANSPOSE
/TABLE FOR SWITCHES
SLROMT=.
/OUTPUT SWITCH
        LROM 4,4,4,4    /Ø = THRU CONNECTION
        LROM 1,2,1,2    /1 = PNPN
        LROM 2,2,2,2    /2 = NNNN
        LROM 1,1,1,1    /3 = PPPP
        LROM 1Ø,2,1Ø,2  /4 = RNRN
        LROM 1,1,2,2    /5 = PPNN
        LROM 2,1Ø,2,1Ø  /6 = NRNR
        Ø               /7 =
/LOWER SWITCH
        LROM 1,1,1,1    /Ø = SELECT SUM
        LROM 2,2,2,2    /1 = SELECT OTHER SUM
        LROM 1Ø,1Ø,1Ø,1Ø        /2 = SELECT JOINT
        LROM 1,4,1,4    /3 = TRANSPOSE AND COMP
        LROM 4,4,4,4    /4 = SELECT COMPLEMENT
        LROM 1,1Ø,1,1Ø  /5 = TRANSPOSE AND INPUT
        LROM 1Ø,1,1,1Ø  /6 = SELECT OLD AND NEW DATA
        Ø               /7 =
/SUM SWITCH
        LROM 4,4,4,4    /Ø = SELECT HALF SUM
        LROM 1Ø,1Ø,1Ø,1Ø        /1 = SELECT LOWER
        LROM 1,1,1,1    /2 = SELECT UPPER
        LROM 2,2,2,2    /3 = SELECT FULL SUM
        LROM 1,4,1,4    /4 = EVEN TO BOTH OUTPUTS
        LROM 1,1Ø,1,1Ø  /5 = SELECT TRANSPOSE
        LROM 1Ø,1,1,1Ø  /6 = SELECT NEW DATA
        LROM 1,1Ø,1Ø,1  /7 = SELECT OLD DATA
/UPPER SWITCH
        LROM 1,1,1,1    /Ø = SELECT SUM
        LROM 2,2,2,2    /1 = SELECT INPUT
        LROM 1Ø,1Ø,1Ø,1Ø        /2 = SELECT JOINT
        LROM 1,1Ø,1Ø,1  /3 = SELECT OLD AND NEW DATA
        LROM 4,4,4,4    /4 = SELECT COMPLEMENT
        Ø               /5
        Ø               /6
        Ø               /7
        .END


PIP V7A

>IOPSØ2 Ø15ØØ7
```

```
BIG R0M C0NTENTS                19-5

CLEA FETL SIZA B0XR TAIC CFEU CLIP NPTM ADTM PSTS BRTU B0LM BRPV
 CFET FETR D0TT L0AE CURV C0MP TANC ZT0S DIVI NSTS BSWU BRPM BWNV
  WAIT M0RE ENDR L0A0 WIN3 C0MQ C0UT PXTM DDIS BWIN BMAS BWNW
   DATA SELF ENDN L0AA SPA1 SETP C0UU NXTM PSTM BEDG BVPT B0LV
    FETC SIZR L0AD L0AR WIN2 EDGE PPTM VIEW NSTM BTRN BDIV BSWV

NAME
0 2 4 6 0 2 4 6 0 2 4 6 0 2 4 6 0 2 4 6 0 2 4 6 0 2 4 6 0 2 4 6 0 2 4 6 USE
 R0UND
..............................1..........1............1.......000003
.........11........1....1.1....1....1..........1..............000008
...................1....1.1....1..1.1..........1..............000007
..............................1...............................000001
.........11........1......1.....1.............................000005
...................1......1.....1.............................000003

 ENABL
.........................................1............1.......000002
..............................1..........1............1.......000003
..............................1...............................000001
..............................................................000000

 ST0RE
....1...........11............................................000003
................1111..........................................000004
............1..11111.1...................................1...1000009
...................1....1......................1..............000003
.1.11..111....1.11.......1......1......11...........11.1..1...000017
........................................1.....................000001
........1.....11...1.1..11......1..............1....1..1.1....000012
........1.....11.....1...1.............1............11.1..1..1000011

 LR0MC
.........11...................................................000002
.1..1...1.....1..........1.............1............11.1..1...000010
...1...1......1...............................................000003
...............11111..1........1..11.......11111.1.....1......000016
.....1.....111.11111.......1.1...1.............111.111...1.1.1000021
......1....1111.......1....1.1.111.............1111.....1...1.000017
..............1...........1.........................11........000004
.1.11..1.1....1....1..1.11.....11......1.......1..............000014
...........1..........1........1....................11........000005
................11..................1...1.....................000004
............1...1.1..1.............1..1.....1.1...............000008
.....11........1...1.1............1.11.....1.1...........1...1000012

 STEPC
..1...........................................................000001
.......1......................................................000001
..............................................................000000
..............................................................000000
...1...1......................................................000002
....................1...........................1.............000002
..............................1...............................000001
.........................................1............1.......000002
.............................1..................1.............000002

 FL0PC
.......................................1......................000001
............................1.................................000001
....................1.........................................000001
1..................1..........................................000002
                                              1...............000001
```

19.5-2

```
1.............1.............1.............1...................
.....................1........................................000001
1.1......................1..1.................................000004

 INPUT
.............................................11...............000002
....................................1.........................000001
.............................................1................000001
..............................................1...............000001
..1...........................................................000001
..1....1......................................................000002
...1...1......................................................000002
..............................................................000000
..............................................................000000
.....11...........................11.11.1..11.................000009
.....11...........................11.11.1..11.................000009
....................................1........11...............000003
....................................1........11...............000003

 PAKEC
....................................1.........................000001
........1......1........................1.....................000003
.......1.......1........................1.....................000003
........1..1..............1...................................000003
..........1...................................................000001
..........................1...................................000001
..........1...................................................000001
.....................................................1......1.000002
.................................................1......1.....000002
..................................................11.......1..000003
.........111............11......11111111.........1...1........000015
.....1..1...............1........................111....1111..000010
.........111..........1..1.....1.................1..1.........000008

 JUMPA
..............................................................000000
..............................................................000000
...............1...............1.1111111..1...................000010
.1..........111......11.11.1..................................000009
.1.....1111....1.....11.1..............1..1...................000011
...11....11.11.1......1.11.1...1..111.........................000015
1..1111........1111.11.1.1.1.....111111...1.....1............1000023
.1.1...1111.11.1............1.1..1.11.11.11.1.1...............000020

 JUMPC
..............1.......1.1......1..........1...................000005
.......11...11.................1..........1...................000006
...1........11........1.1.................1...................000006
..............................................................000000
...1...11...111.......1.1...1.11.........11.1.1...............000015
11..111..11....1111.11.1.1.1.....1111111........1............1000025
..............................1..........1............1.......000003
..............................................................000000
.1............................1.................1.............000003
1..11111111.1111111.1.1111.11......11111.11.1.1..............1000033
...1.........1.......1...1....1...............................000005
.............11....................1...1........1.............000005
....11...................................1..1.................000004
...............................1.11...........................000003
....11.......11................1..1...........................000006
...11..................................1......................000003
.1...................1.........1.........1....................000004
.........................1.......1.1.....1..1.................000005

MORE LINE DRIVE REQUIRED
```

19.5-3

```
CLEA ØØØØØ6
CFET ØØØØØ9
WAIT ØØØØØ4
DATA ØØØØ13
FETC ØØØØ11
FETL ØØØØ1Ø
FETR ØØØØØ7
MØRE ØØØØ13
SELF ØØØØ12
SIZR ØØØØ12
SIZA ØØØØ12
DØTT ØØØØØ6
ENDR ØØØØ11
ENDN ØØØØ12
LØAD ØØØØ15
BØXR ØØØØ15
LØAE ØØØØ11
LØAØ ØØØØ1Ø
LØAA ØØØØØ8
LØAR ØØØØ13
TAIC ØØØØØ5
CURV ØØØØ12
WIN3 ØØØØ12
SPA1 ØØØØØ3
WIN2 ØØØØ14
CFEU ØØØØ15
CØMP ØØØØØ7
CØMQ ØØØØØ7
SETP ØØØØØ6
EDGE ØØØØØ3
CLIP ØØØØ1Ø
TANC ØØØØ15
CØUT ØØØØØ7
CØUU ØØØØØ9
PPTM ØØØØ12
NPTM ØØØØ13
ZTØS ØØØØ15
PXTM ØØØØØ8
NXTM ØØØØØ9
VIEW ØØØØ13
ADTM ØØØØØ7
DIVI ØØØØ11
DDIS ØØØØ1Ø
PSTM ØØØØØ4
NSTM ØØØØØ9
PSTS ØØØØØ6
NSTS ØØØØ1Ø
BWIN ØØØØØ8
BEDG ØØØØØ8
BTRN ØØØØØ7
BRTU ØØØØØ3
BSWU ØØØØØ3
BMAS ØØØØØ8
BVPT ØØØØØ8
BDIV ØØØØØ5
BØLM ØØØØØ5
BRPM ØØØØØ3
BWNW ØØØØØ5
BØLV ØØØØØ4
BSWV ØØØØØ3
BRPV ØØØØØ2
BWNV ØØØØØ7
```

19.6

```
        .TITLE CLIPBA
/THIS TAPE DEFINES THE COMPONENT PLACEMENT FOR THE CLIPPER ROM
/THESE MACROS SET UP TABLES FOR ROM ASSIGNMENTS
/THE 4ØØØ BIT SAYS "THIS IS THE FINAL OUTPUT OF THE BIT LINE"
/THE INCREMENTS Ø,4,1Ø,14 ARE APPLIED TO THE INPUTS TO THE
/FOUR-INPUT "OR" GATE WHICH PRODUCES THE BIT LINE OUTPUT.
/EXTENSIONS ARE HANDLED BY THE MACROS "CONT", "CONA" AND "CONB".
        .DEFIN BUG2,W
        W+4ØØØ
        W-4
        W
        4ØØØØØ
        .ENDM
        .DEFIN BUG4,W
        W+4ØØØ
        W
        W+4
        W+1Ø
        W+14
        4ØØØØØ
        .ENDM
        .DEFIN BUG7,W,E
        W+4ØØØ
        W
        W+4
        W+1Ø
        W+2Ø14
        CONT E
        4ØØØØØ
        .ENDM
        .DEFIN BUG8,W,E,F
        W+4ØØØ
        W-4+2ØØØ
        CONT E
        W+2ØØØ
        CONT F
        4ØØØØØ
        .ENDM
        .DEFIN BUG1Ø,W,E,F
        W+4ØØØ
        W
        W+4
        W+2Ø1Ø
        CONT E
        W+2Ø14
        CONT F
        4ØØØØØ
        .ENDM
        .DEFIN BUG13,W,E,F,G
        W+4ØØØ
        W
        W+2ØØ4
        CONT E
        W+2Ø1Ø
        CONT F
        W+2Ø14
        CONT G
        4ØØØØØ
        .ENDM
```

```
.DEFIN BUG16,W,E,F,G,H
W+4000
W+2000
CONT E
W+2004
CONT F
W+2010
CONT G
W+2014
CONT H
400000
.ENDM
.DEFIN BUG19,W,E,F,G,H,J
W+4000
W+2000
CONA E,F
W+2004
CONT G
W+2010
CONT H
W+2014
CONT J
400000
.ENDM
.DEFIN BUG22,W,E,F,G,H,J,K
W+4000
W+2000
CONA E,F
W+2004
CONA G,H
W+2010
CONT J
W+2014
CONT K
400000
.ENDM
.DEFIN BUG25,W,A,B,C,D,E,F,G
W+4000  /OUTPUT
W+2000  /FIRST INPUT TO THE OR GATE
CONA A,B         /PUT INPUTS 1-8 ON EXTENDERS
W+2004
CONA C,D
W+2010
CONA E,F
W+2014  /LAST INPUT
CONT G
400000
.ENDM
.DEFIN BUG28,W,A,B,C,D,E,F,G,H
W+4000
W+2000
CONA A,B
W+2004
CONA C,D
W+2010
CONA E,F
W+2014
CONA G,H
400000
.ENDM
```

```
        .DEFIN BUG34,W,A,B,C,D,E,F,G,H,I,J
        W+4ØØØ
        W+2ØØØ
        CONB A,B,C
        W+2ØØ4
        CONB D,E,F
        W+2Ø1Ø
        CONA G,H
        W+2Ø14
        CONA I,J
        4ØØØØØ
        .ENDM
        .DEFIN BUG37,W,A,B,C,D,E,F,G,H,I,J,K
        W+4ØØØ
        W+2ØØØ
        CONB A,B,C
        W+2ØØ4
        CONB D,E,F
        W+2Ø1Ø
        CONB G,H,I
        W+2Ø14
        CONA J,K
        4ØØØØØ
        .ENDM
        .DEFIN CONT,W
        W+1ØØØ
        W
        W+4
        W+1Ø
        W+14
        .ENDM
        .DEFIN CONA,W,E
        W+1ØØØ
        W
        W+4
        W+1Ø
        W+2Ø14
        CONT E
        .ENDM
        .DEFIN CONB,W,B,C
        W+1ØØØ
        W
        W+4
        W+2Ø1Ø
        CONT B
        W+2Ø14
        CONT C
        .ENDM
        .DEFIN BIT,N,A,B
        Ø
        .ASCII +N+
        A
        B
        .ENDM
/BUG SYMBOL EQUALITIES
/VALUE OF LAST TWO BITS TELL THE COLUMN NUMBER
        /ØØ=A COLUMN (INNER RIGHT)
        /Ø1=B COLUMN (OUTER LEFT)
        /1Ø=C COLUMN (OUTER RIGHT)
        /11=D COLUMN (INNER LEFT)
```

```
A=Ø
B=1
C=2
D=3
E=2Ø
F=21
G=22
H=23
J=4Ø
K=41
L=42
M=43
N=6Ø
P=61
Q=62
R=63
S=1ØØ
T=1Ø1
U=1Ø2
V=1Ø3
W=12Ø
X=121
Y=122
Z=123
AA=14Ø
BB=141
CC=142
DD=143
EE=16Ø
FF=161
GG=162
HH=163
JJ=2ØØ
KK=2Ø1
LL=2Ø2
MM=2Ø3
EEU=164
EEL=174
FFU=165
FFL=175
GGU=166
GGL=176
HHU=167
HHL=177
JJU=2Ø4
JJL=214
KKU=2Ø5
KKL=215
LLU=2Ø6
LLL=216
MMU=2Ø7
MML=217
        .GLOBL BOARD1,BOARD2,BOARD3,BOARD4
        .GLOBL BOARD5,BOARD6,BOARD7,BOARD8
/HERE ARE THE BOARD DEFINITIONS
BOARD1 Ø        /TENS DIGIT
        3       /UNITS DIGIT
        BIT RND1,4Ø,1
        BUG4 KK
        BIT RND2,2Ø,1
```

19.6 - 5

```
        BUG8 HHL,B,D
        BIT RND3,1Ø,1
        BUG7 BB,T
        BIT RND4,4,1
        BUG2 FFU
        BIT RND5,2,1
        BUG7 DD,V
        BIT RND6,1,1
        BUG4 X
/ENABLE BITS
        BIT ENA1,1Ø,2
        BUG2 FFL
        BIT ENA2,4,2
        BUG4 Z
        BIT ENA3,2,2
        BUG2 HHU
        /ENA4 IS NOT USED
/STORE BITS
        BIT STR1,2ØØ,3
        BUG4 AA
        BIT STR2,1ØØ,3
        BUG4 W
        BIT STR3,4Ø,3
        BUG1Ø Y,C,E
        BIT STR4,2Ø,3
        BUG4 CC
        BIT STR5,1Ø,3
        BUG19 MM,F,H,K,M,P
        BIT STR6,4,3
        BUG2 EEL
        BIT STR7,2,3
        BUG13 JJ,G,J,L
        BIT STR8,1,3
        BUG13 LL,N,Q,A
        BIT STE1,4ØØ,5
        BUG2 GGU
        BIT STE2,2ØØ,5
        BUG2 GGL
        /STE3 NOT USED
                  /STE4 NOT USED
        BIT STE5,2Ø,5
        BUG2 EEU
        BIT STE6,1Ø,5
        BUG4 U
        4ØØØØ1
/END OF FIRST BOARD
BOARD2 Ø        /TENS DIGIT
       4        /UNITS DIGIT
        BIT LRO1,4ØØØ,4
        BUG2 EEU
        BIT LRO2,2ØØØ,4
        BUG1Ø MM,D,B
        BIT LRO3,1ØØØ,4
        BUG4 JJ
        BIT LRO4,4ØØ,4
        BUG16 LL,A,C,E,G
        BIT LRO5,2ØØ,4
        BUG22 FF,K,M,P,R,T,V
        BIT LRO6,1ØØ,4
        BUG19 AA,J,L,N,Q,S

>
```

```
        BIT LRO7,4Ø,4
        BUG4 HH
        /LRO3 ELSEWHERE
        BIT LRO9,1Ø,4
        BUG7 Y,U
        BIT LR1Ø,4,4
        BUG4 CC
        BIT LR11,2,4
        BUG1Ø KK,F,H
        /LR12 ELSEWHERE
        BIT STE7,4,5
        BUG2 EEL
        BIT STE8,2,5
        BUG2 GGU
        BIT STE9,1,5
        BUG2 GGL
        4ØØØØ1
/END OF SECOND BOARD
BOARD3 Ø          /TENS DIGIT
        5         /UNITS DIGIT
        /JUA1 NOT USED
        /JUA2 NOT USED
        BIT JUA3,4Ø,11
        BUG13 KK,B,D,F
        BIT JUA4,2Ø,11
        BUG1Ø X,K,M
        BIT JUA5,1Ø,11
        BUG13 AA,A,C,E
        BIT JUA6,4,11
        BUG16 MM,P,R,T,V
        /JUA7 ELSEWHERE
        BIT JUA8,1,11
        BUG22 CC,G,J,L,N,Q,S
        BIT PKCØ,1ØØØØ,1Ø
        BUG2 FFU
        BIT FLO1,2ØØ,6
        BUG2 HHU
        BIT FLO2,1ØØ,6
        BUG2 FFL
        BIT FLO3,4Ø,6
        BUG2 HHL
        BIT FLO4,2Ø,6
        BUG2 EEU
        BIT FLO5,1Ø,6
        BUG2 GGU
        BIT STE4,4Ø,5
        BUG2 GGL
        BIT FLO6,4,6
        BUG4 DD
        BIT FLO7,2,6
        BUG2 EEL
        BIT FLO8,1,6
        BUG4 BB
        BIT INO1,4ØØØ,7
        BUG4 W
        BIT INO2,2ØØØ,7
        BUG4 Y
        BIT INO3,1ØØØ,7
        BUG4 JJ
        BIT INO4,4ØØ,7
```

19.6-7

```
        BUG4 LL
        4ØØØØ1

BOARD4  Ø         /TENS
        6         /UNITS
        BIT PK1Ø,4,1Ø
        BUG16 BB,B,D,F,H
        BIT PK11,2,1Ø
        BUG13 DD,K,M,R
        BIT PK12,1,1Ø
        BUG1Ø X,T,V
        /PKC1 ELSEWHERE
        /PKC2 ELSEWHERE
        BIT PKC3,1ØØØ,1Ø
        BUG4 Z
        BIT PKC4,4ØØ,1Ø
        BUG2 KKL
        BIT PKC5,2ØØ,1Ø
        BUG2 MMU
        BIT PKC6,1ØØ,1Ø
        BUG2 MML
        BIT PKC7,4Ø,1Ø
        BUG2 KKU
        BIT PKC8,2Ø,1Ø
        BUG4 HH
        /PKC9 ELSEWHERE
        BIT JUC2,2ØØØØØ,13
        BUG7 JJ,A
        BIT JUC3,1ØØØØØ,13
        BUG7 AA,C
        /JUC4 NOT USED
        BIT JUC5,2ØØØØ,13
        BUG16 CC,J,L,N,Q
        BIT JU11,2ØØ,13
        BUG7 EE,S
        BIT JU12,1ØØ,13
        BUG7 GG,U
        BIT JU15,1Ø,13
        BUG7 FF,P
        4ØØØØ1

BOARD5  Ø
        7
        BIT JU1Ø,4ØØ,13
        BUG34 AA,A,C,E,G,J,L,N,Q,S,U
        BIT JUC7,4ØØØ,13
        BUG4 CC
        /JUC8 NOT USED
        BIT JUC9,1ØØØ,13
        BUG4 EE
        BIT JU13,4Ø,13
        BUG4 GG
        BIT JU16,4,13
        BUG4 LL
        BIT JU17,2,13
        BUG4 JJ
        BIT JUC6,1ØØØØ,13
        BUG25 HH,B,D,F,H,K,M,P
        BIT JUC1,4ØØØØØ,13
        BUG7 MM,R
```

```
                /JUC4 NOT USED
        BIT JU14,2Ø,13
        BUG4 KK
        BIT PKC1,4ØØØ,1Ø
        BUG4 FF
        BIT PKC2,2ØØØ,1Ø
        BUG4 DD
        BIT PKC9,1Ø,1Ø
        BUG4 BB
        4ØØØØ1

BOARD6 Ø
        1Ø      /CARD 8
        BIT JUA7,2,11
        BUG25 FF,B,D,F,H,K,M,P
        BIT LR12,1,4
        BUG13 HH,R,T,V
        BIT JU18,1,13
        BUG7 CC,C
        BIT INO5,2ØØ,7
        BUG2 JJU
        BIT INO6,1ØØ,7
        BUG2 LLU
        BIT INO3,1ØØØØ,7
        BUG2 JJL
        BIT INO8,2Ø,7
        BUG2 LLL
        BIT INO9,1Ø,7   /ALSO BIT 1Ø
        BUG1Ø EE,N,A
        BIT IN11,2,7    /ALSO BIT 12
        BUG4 GG
        BIT LRO8,2Ø,4
        BUG16 AA,E,G,J,L
        4ØØØØ1
BOARD7
BOARD8
        .END

END OF FILE REACHED BY:
P 6Ø
>
```