H-79-0350

Graphic 7™

COMPUTER GRAPHICS DISPLAY SYSTEM

MODEL 5753 2-D/3-D COORDINATE CONVERTER USER'S MANUAL

Information Products Division
Federal Systems Group
SANDERS
Daniel Webster Highway, South — Nashua, New Hampshire 03061

H-79-0350

Graphic 7™

COMPUTER GRAPHICS DISPLAY SYSTEM

MODEL 5753 2-D/3-D COORDINATE CONVERTER USER'S MANUAL

Information Products Division
Federal Systems Group
SANDERS
Daniel Webster Highway, South — Nashua, New Hampshire 03061

Sanders Associates, Inc., reserves the right to modify the products described in this manual and to make corrections or alterations to this manual at any time without notice.

First Edition - May 1980
Reprint - June 1980
Reprint - October 1980
Reprint - March 1981
Reprint - July 1981 - Change 1
Reprint - October 1981

RECORD OF CHANGES

CHANGE NO.	DATE	TITLE OR BRIEF DESCRIPTION	ENTERED BY
1	June 81	Adds 2D programming	

TABLE OF CONTENTS

Paragraph Page

SECTION 1

INTRODUCTION

This User's Manual describes the operational modes of the Model 5753 2-D/3-D coordinate converter, its instruction set, its parameter and control registers, and its interrupts. This manual also describes the use of the 2-D/3-D Coordinate Converter via the associated GCP+ instructions and FSP subroutines.

SECTION 2

OPERATIONAL MODES

The coordinate converter has the following operational modes:

2D or 3D
Homogeneous or non-homogeneous
Normal or PHOTOPEN mode
Clipping or no clipping
Perspective or non-perspective

The desired modes are set in the dimension register. They determine how the coordinate converter processes the user's image file. Processing takes place within the coordinate converter in the following order:

Coordinate transformation
Homogeneous conversion (optional)
Perspective generation (optional)
Clipping (optional)
Refresh code generation

2.1 COORDINATES

Object points may be defined in 2D or 3D coordinates, X, Y or X,Y,Z respectively. The range of coordinates in X and Y is ±32K. The range of coordinates in Z is 0 to +32K, where 0 is in the same plane as the screen and positive Z increases with depth into the screen. Thus, the coordinates are operated upon in a left-handed system. Negative values of Z are not permitted for coordinates. The range of coordinates in X, Y, and Z define the usable image space.

Coordinates are expressed in two's complement notation.

2.2 EYE POSITION

When using 3D with perspective, you must specify the observer's eye position, Xa, Ya, Za. Xa and Ya can range from +32K to -32K. Za ranges from 0 to -32K, where 0 is in the same plane as the screen and Z increases in the negative sense as you move out of the screen.

PHOTOPEN is a trademark of Sanders Associates, Inc.

2.3 COORDINATE TRANSFORMATION

Coordinate transformation is the concatenation of a coordinate point by a composite matrix. The composite matrix contains any desired combination of scaling, rotation, and translation. Its derivation is discussed later.

The form of the actual concatenation is shown below for each operating mode.

2D Non-homogeneous

$$\begin{bmatrix} X_1, & Y_1 \end{bmatrix} = \begin{bmatrix} X_0, & Y_0 \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \\ M_{31} & M_{32} \end{bmatrix}$$

2D Homogeneous

$$\begin{bmatrix} X_1, & Y_1, & W_1 \end{bmatrix} = \begin{bmatrix} X_0, & Y_0, & W_0 \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}$$

3D Non-homogeneous

$$\begin{bmatrix} X_1, & Y_1, & Z_1 \end{bmatrix} = \begin{bmatrix} X_0, & Y_0, & Z_0 \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \\ M_{41} & M_{42} & M_{43} \end{bmatrix}$$

3D Homogeneous

$$\begin{bmatrix} X_1, & Y_1, & Z_1, & W_1 \end{bmatrix} = \begin{bmatrix} X_0, & Y_0, & Z_0, & W_0 \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix}$$

2.4 HOMOGENEOUS MODE

Using homogeneous mode involves an additional coordinate, W, which modifies the other coordinates of each point. Thus, 2D homogeneous points have 3 coordinates, X, Y, and W; 3D homogeneous points have 4 coordinates, X, Y, Z, and W.

Dehomogeneization is the division of W into X, Y, and Z (or just X and Y) as shown below.

2D

$$\begin{bmatrix} X, & Y, & W \end{bmatrix} \rightarrow \begin{bmatrix} X/W, & Y/W \end{bmatrix}$$

3D

$$\begin{bmatrix} X, & Y, & Z, & W \end{bmatrix} \rightarrow \begin{bmatrix} X/W, & Y/W, & Z/W \end{bmatrix}$$

W is expressed as a fraction with limits:

$$0.0 < W \leq 1.0$$

Therefore, it can be seen that W can be used for scaling. As W decreases in size, the resulting scaled values are larger.

2.5 PERSPECTIVE

Perspective application is a 3D function in which X and Y coordinate values are modified as a function of their depth (Z coordinate value) and of the observer's eye position (X_a, Y_a, Z_a).

Figures 2-1 through 2-3 illustrate the relationship between the eye position, coordinate points, and the screen in 3-D space. In these figures, two identical 3-dimensional cubes have been set up in 3-D space. X_a and Y_a have been set at 0 for simplicity. Z_a has been set to an arbitrary value which places both cubes in the viewable area.

In figure 2-1, the large square represents the viewable area of the screen. The smaller squares represent the cubes, shown with no perspective (orthographic projection). Figures 2-2 and 2-3 show the top and right side views, respectively. The dark line in the center is the actual screen, situated at Z = 0. The point to the left where the lines converge is the eye position. The large wedge which extends from the eye position through the ends of the screen line defines the actual viewable area. Any coordinate points which are within this wedge and to the right of the screen (positive Z) can be viewed from the defined eye position. Each viewable coordinate point can be mapped onto the screen at the intersection of the screen and the line which extends from the eye position to the coordinate point.

Figure 2-4 shows the result of this mapping. It shows the same objects as figure 2-1 with perspective, as seen from the eye position.

The coordinate converter, when operating in perspective mode, automatically performs the mapping between the 3-dimensional viewing area as seen from the user defined eye position and the 2-dimensional screen.

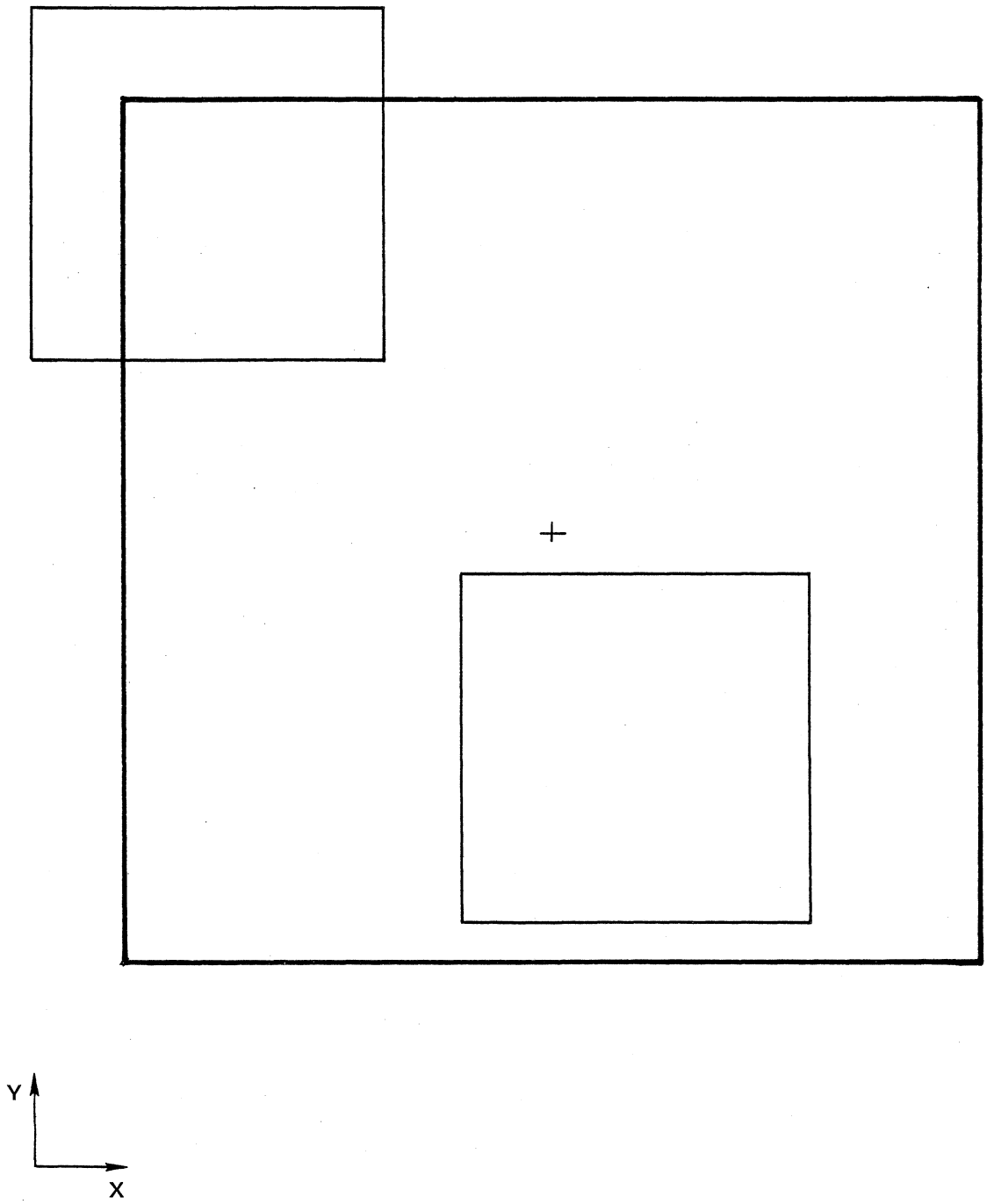


Figure 2-1

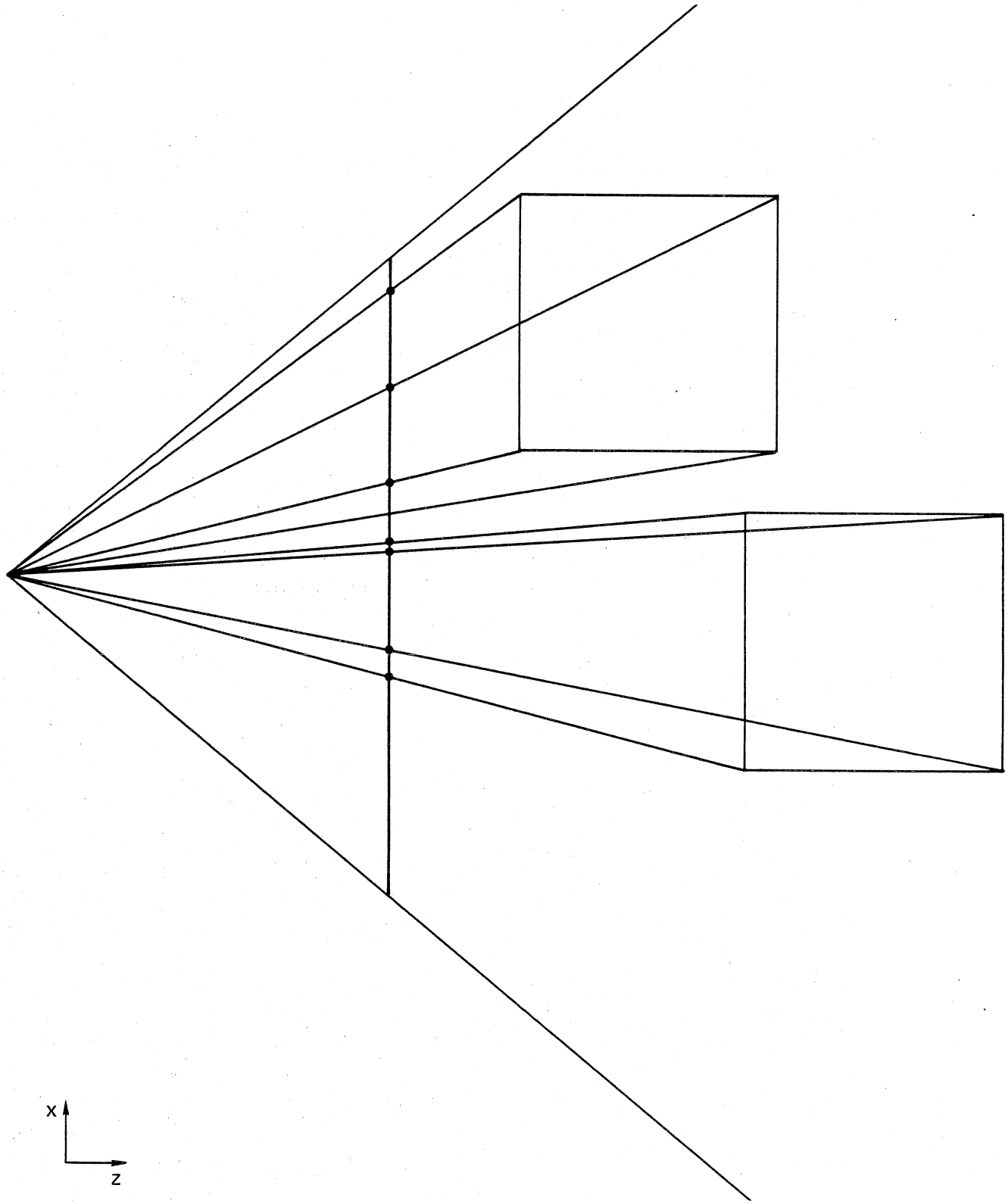


Figure 2-2

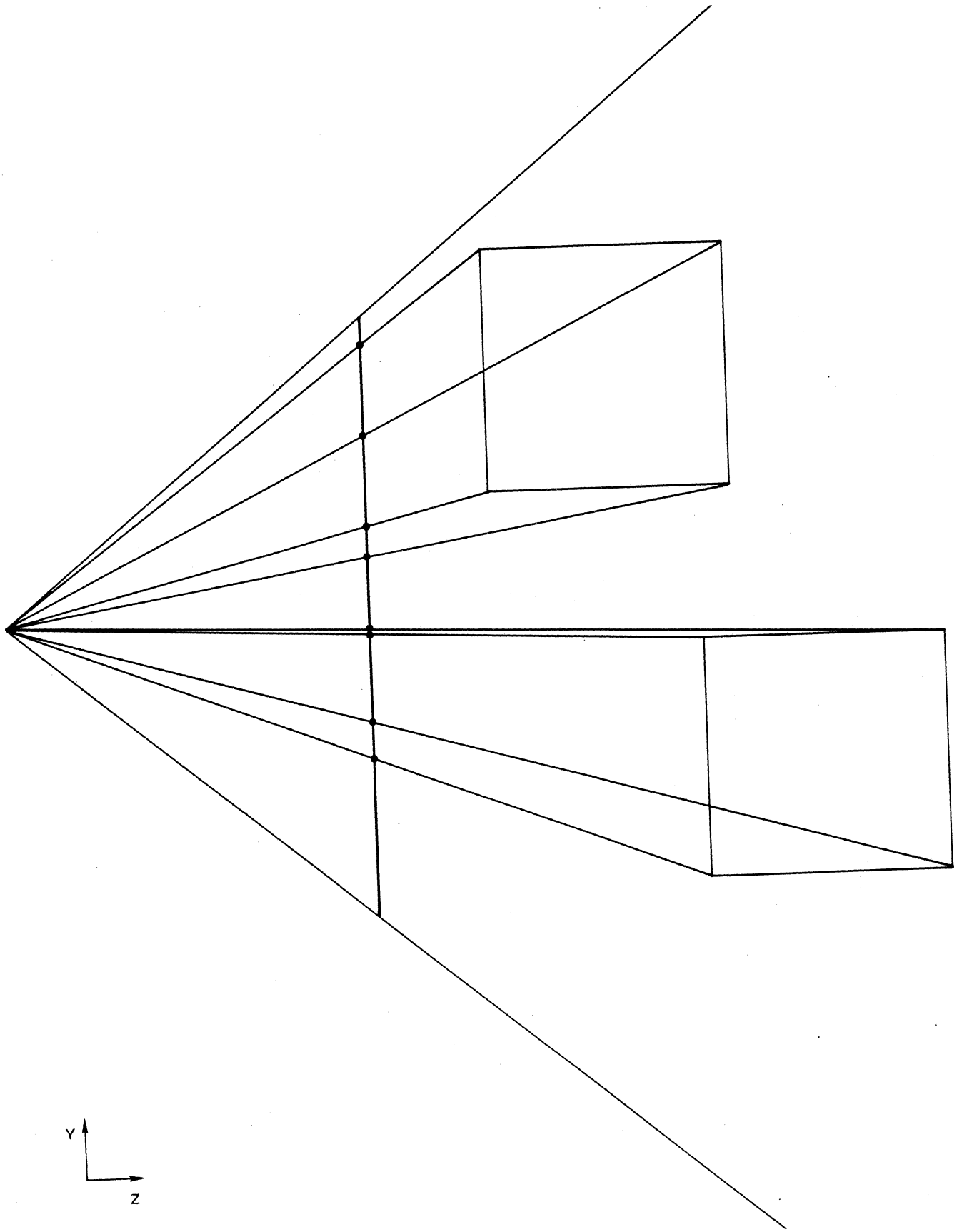


Figure 2-3

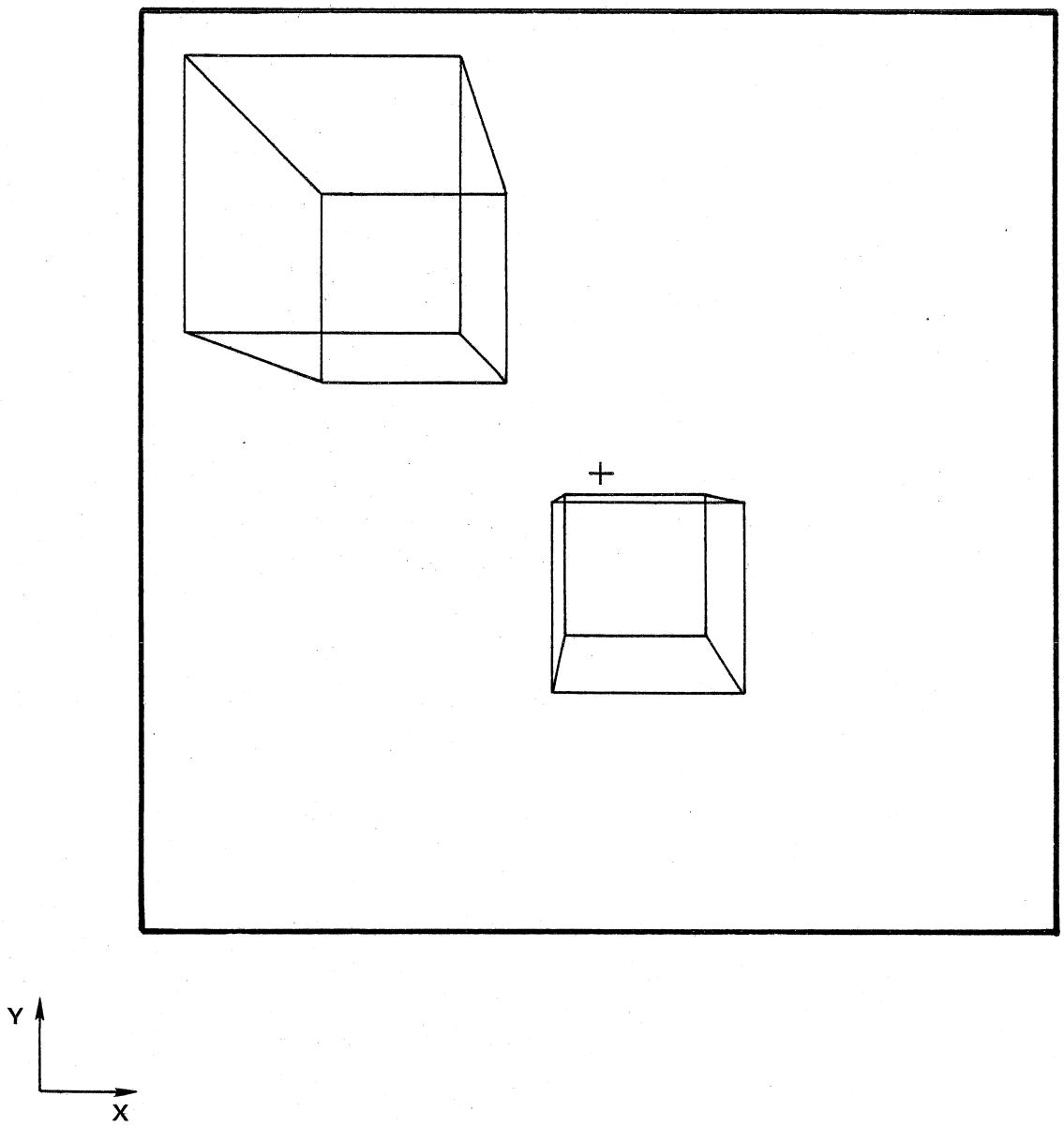


Figure 2-4

2.6 CLIPPING

Clipping is a coordinate converter function which eliminates graphic data that falls outside a user-definable 2D or 3D box. The clipping box or viewbox is defined in terms of six boundary planes (four for 2D): left, right, top, bottom, near and far (Lv, Rv, Tv, Bv, Nv, Fv). Vectors that are totally outside the clipping box are rejected. Vectors which cross a clipping boundary are clipped; that is, the portion of the vector which falls outside of the clipping box is eliminated. When using perspective, coordinate points are modified for perspective before comparison to the clipping boundaries. That is, the user-defined viewbox is defined with perspective already applied. Characters drawn with graphic controller instructions are also clipped. However, vectors drawn with graphic controller instructions are not clipped.

2.7 REFRESH GENERATION

The coordinate converter translates certain 3D instructions into refresh code recognizable by the graphic controller. Prior to this translation, you must specify where the refresh code is to be placed in memory. Note that the refresh code cannot cross a 32K word boundary. The lower 16 bits of the starting address should be placed in the refresh address register. The upper 2 bits (which specify the 32K block number) should be placed in bits 9 and 8 of the block register.

If you want to partition refresh memory into blocks, then you should place the address of the end of the block in the refresh limit register. The refresh used-up bit in the mask register should also be enabled. When the coordinate converter reaches the end of the block while generating refresh code, it sets the refresh used-up bit in the status register and halts. This causes a refresh used-up interrupt to be sent to the display processor. In response to this interrupt, you should write the starting address of the next block into the refresh address register and the ending address of the new block into the refresh limit register. You should then access the continue register to restart the coordinate converter. The coordinate converter then inserts a graphic controller relative jump instruction into the previous refresh block to provide the linkage to the new refresh block. It then continues processing as before.

2.8 PEN MODE

Pen mode is used to associate a word of refresh code, which was previously generated by the coordinate converter, to the coordinate converter instruction which was translated into that word of refresh code. This feature can be used in conjunction with a PHOTOPEN to relate a displayed vector to the coordinate converter instruction which generated it.

Before starting a pen mode search, it is necessary to set the PHOTOPEN match bit in the mask register and to load the PHOTOPEN strike address register with the address of the word of refresh code to search for. Except for setting the pen mode bit in the dimension register, the image file should appear exactly as it did when the target refresh word was actually generated.

Pen mode processing differs from normal mode processing in the following way. Instead of writing refresh code into memory, the coordinate converter compares the address of the location where it would normally write each word of refresh to the contents of the PHOTOPEN strike address register. When a match occurs, the coordinate converter sets the PHOTOPEN match bit in the status register, which causes an interrupt to be sent to the display processor.

In response to this interrupt, you should read the program counter to determine the address in the image file where the match occurred. Note that the program counter has already been updated by 2 bytes at this point.

If refresh memory has been partitioned into blocks, you will not receive any refresh used-up interrupts. The coordinate converter will go from block to block by reading the jump relative instruction which has previously been inserted at the end of each block. Note that refresh blocks must have a uniform size.

SECTION 3

COORDINATE CONVERTER INSTRUCTIONS

The coordinate converter instruction set consists of 41 unique instructions plus all of the existing graphic controller instructions. The unique instructions comprise three basic categories: refresh control instructions, sequence control instructions, and parameter instructions. Refresh control instructions define the object to be converted and translated into refresh code. Sequence control instructions specify the sequence of program execution by using jumps and calls. Parameter instructions allow modification of variables which affect the operation of the coordinate converter.

3.1 OPERAND ADDRESSING

Most of the refresh control and parameter instructions can be used with three different addressing modes: immediate, deferred, and deferred relative. When using immediate mode, instruction operands follow directly in line after the instruction op code. With deferred mode, the contents of the word following the op code is the address of the first operand. Any additional operands follow the first in consecutive words. With deferred relative mode, the contents of the word following the op code is the relative address of the first operand. This relative address is added to the program counter (which has been updated and now contains the address of the instruction op code plus 4 bytes) and the result is used to address the operand. Both deferred and deferred relative addresses must be in even bytes. In the deferred cases, after the operands have been loaded, the next instruction is fetched from the word following the word containing the deferred address.

3.2 REFRESH CONTROL INSTRUCTIONS

Refresh control instructions consist of moves and draws which define the image to be transformed by the matrix parameters and translated into graphic controller refresh code.

The moves and draws define points through coordinate blocks which vary in length from 2 to 4 words, made up of X,Y,Z, and W coordinates, Z and W being optional depending on the dimension (2D/3D, homogeneous/non-homogeneous). The actual coordinates can be in terms of absolute or relative position. Each move and draw instruction is available in all three addressing modes.

When clipping is enabled, several differences can be noted about moves and draws. For points which are outside the clipping window, refresh code generation is inhibited. For vectors which cross clipping boundaries, refresh code is generated to account for boundary point intersections.

AMV3 MOVE ABSOLUTE Octal Code: 006404

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	2D		3D	
0	0	0	0	1	1	0	1	0	0	0	0	0	1	0	0	N-H	H	N-H	H
X coordinate																X	X	X	X
Y coordinate																X	X	X	X
Z coordinate																		X	X
W coordinate																	X		X

The AMV3 instruction converts the point, defined by the variable length coordinate block, through the transformation defined by the previously loaded matrix parameters. The converted point is then translated into the following refresh code and loaded into memory at the location specified by the refresh register.

LDXA Load X absolute

MVYA Move Y absolute

AMD3 MOVE ABSOLUTE DEFERRED Octal Code: 006444

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	1	0	0	1	0	0
Address of coordinate block															

Same as AMV3 except that second word of instruction is pointer to variable length coordinate block.

AMDR MOVE ABSOLUTE DEFERRED RELATIVE Octal Code: 006464

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	1	1	0	1	0	0
Relative address of coordinate block															

Same as AMV3 except that second word of instruction is relative address of variable length coordinate block.

RMV3 MOVE RELATIVE Octal Code: 006405

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	0	0	0	1	0	1
X coordinate															
Y coordinate															
Z coordinate															
W coordinate															

2D		3D	
N-H	H	N-H	H
X	X	X	X
X	X	X	X
		X	X
	X		X

The RMV3 instruction adds the relative coordinates, defined by the variable length coordinate block, to the last defined point and converts the result through the transformation defined by the previously loaded matrix parameters. The converted point is then translated into the following refresh code and loaded into memory at the location specified by the refresh address register.

LDXA Load X absolute

MVYA Move Y absolute

RMD3 MOVE RELATIVE DEFERRED Octal Code: 006445

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	1	0	0	1	0	1
Address of coordinate block															

Same as RMV3 except that second word of instruction is pointer to variable length coordinate block.

RMDR MOVE RELATIVE DEFERRED RELATIVE Octal Code: 006465

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	1	1	0	1	0	1
Relative address of coordinate block															

Same as RMV3 except that second word of instruction is relative address of variable length coordinate block.

ADR3 DRAW ABSOLUTE Octal Code: 006406

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	0	0	0	1	1	0
X coordinate															
Y coordinate															
Z coordinate															
W coordinate															

2D		3D	
N-H	H	N-H	H
X	X	X	X
X	X	X	X
		X	X
	X		X

The ADR3 instruction converts the point, defined by the variable length coordinate block, through the transformation defined by the previously loaded matrix parameters. The converted point is then translated into the following refresh code and loaded into memory at the location specified by the refresh address register.

LDXA Load X absolute

DRYA Draw Y absolute

ADD3 DRAW ABSOLUTE DEFERRED Octal Code: 006446

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	1	0	0	1	1	0
Address of coordinate block															

Same as ADR3 except that second word of instruction is pointer to variable length coordinate block.

ADDR DRAW ABSOLUTE DEFERRED RELATIVE Octal Code: 006466

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	1	1	0	1	1	0
Relative address of coordinate block															

Same as ADR3 except that second word of instruction is relative address of variable length coordinate block.

RDR3 DRAW RELATIVE Octal Code: 006407

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	0	0	0	1	1	1
X coordinate															
Y coordinate															
Z coordinate															
W coordinate															

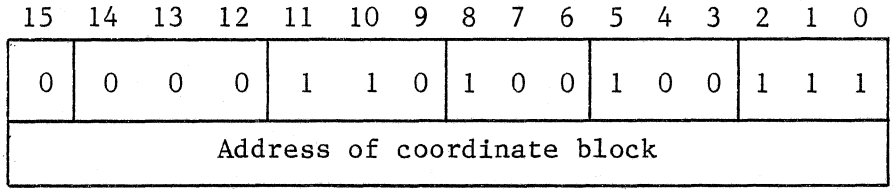
2D		3D	
N-H	H	N-H	H
X	X	X	X
X	X	X	X
		X	X
	X		X

The RDR3 instruction adds the relative coordinates, defined by the variable length coordinate block to the last defined point and converts the result through the transformation defined by the previously loaded matrix parameters. The converted point is then translated into the following refresh code and loaded into memory at the location specified by the refresh address register.

LDXA Load X absolute

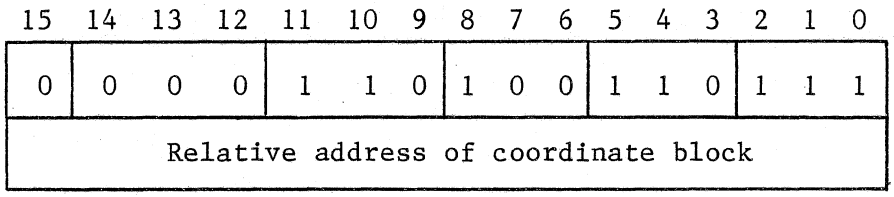
DRYA Draw Y absolute

RDD3 DRAW RELATIVE DEFERRED Octal Code: 006447



Same as RDR3 except that second word of instruction is pointer to variable length coordinate block.

RDDR DRAW RELATIVE DEFERRED RELATIVE Octal Code: 006467

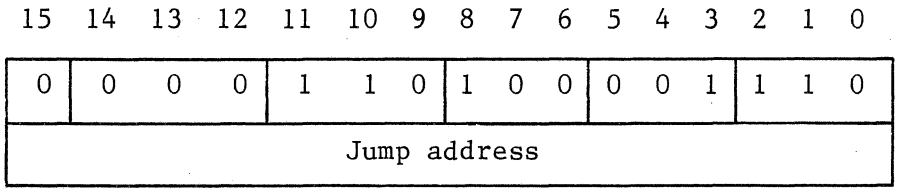


Same as RDR3 except that second word of instruction is relative address of variable length coordinate block.

3.3 SEQUENCE CONTROL INSTRUCTIONS

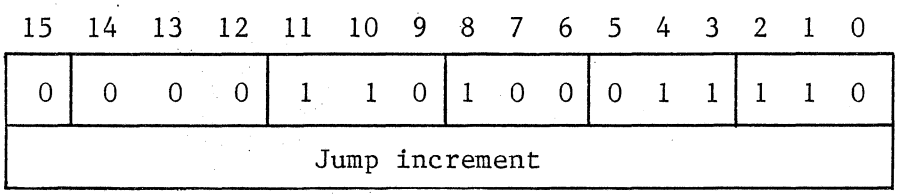
These instructions are used to unconditionally control the sequence of program execution by the coordinate conveter.

JP3A — JUMP ABSOLUTE — Octal Code: 006416



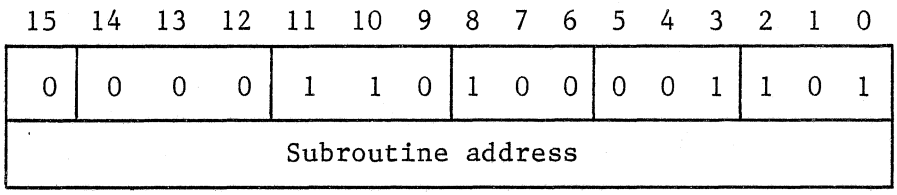
JP3A is a 2-word instruction which transfers program control to the absolute address specified in the second word of the instruction.

JP3R — JUMP RELATIVE — Octal Code: 006436



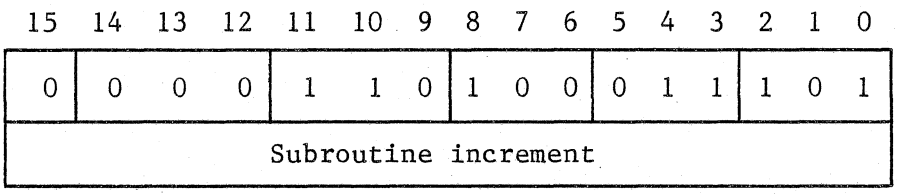
JP3R is a 2-word instruction which transfers program control to a relative memory location. The content of the second word of the instruction is added to the program counter, which is pointing to the address following the jump increment. The result is then used as the address of the next instruction to be executed.

CL3A CALL SUBROUTINE ABSOLUTE Octal Code: 006415



CL3A is a 2-word instruction which calls a subroutine whose address is specified by the second word of the instruction. When the instruction is executed, the content of the program counter (which is pointing to the address following the location of the subroutine address) is pushed onto the coordinate converter stack. This saves the address of the instruction to be executed following completion of the subroutine. The content of the second word of the instruction is then loaded into the program counter and used as the address of the next instruction to be executed.

CL3R CALL SUBROUTINE RELATIVE Octal Code: 006435



CL3R is a 2-word instruction which calls a subroutine from a relative memory location. When the instruction is executed, the content of the program counter (which points to the address following the location of the subroutine increment) is pushed onto the coordinate converter stack. This saves the address of the instruction to be executed following completion of the subroutine. The content of the second word of the instruction is then added to the program counter and the result is used as the address of the next instruction to be executed.

RTN3 RETURN Octal Code: 006420

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0

RTN3 is normally the last instruction of a subroutine and causes program control to return to the calling program. When the instruction is executed, the content of the location pointed to by the coordinate converter stack pointer is popped from the stack, loaded into the program counter, and used as the address of the next instruction to be executed.

NOP3 NO OPERATION Octal Code: 006400

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0

NOP3 is generally a filler instruction which does not perform any action.

HLT3 HALT Octal Code: 006422

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	0	1	0	0	1	0

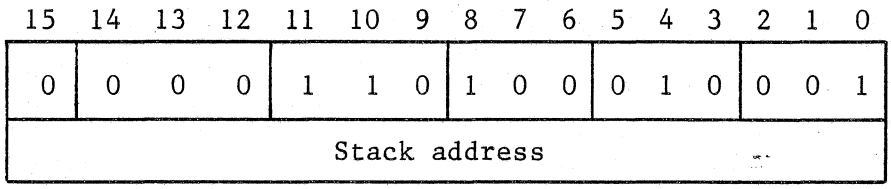
The HLT3 instruction should always be the last instruction in the image file. It causes the coordinate converter to set the halt bit in the status register before going into the halt state. This also causes a halt interrupt to be sent to the display processor, if enabled.

3.4 PARAMETER INSTRUCTIONS

Parameter instructions are used to establish or change various parameters which affect the operation of the coordinate converter. Note that all of these parameters can also be modified using display processor instructions to write to the individual parameter registers (see Section 5).

With the exception of LSP3, all of these parameter instructions are available in all three addressing modes. In each instruction the associated parameters are loaded into the indicated coordinate converter registers. See Section 5 for descriptions of the individual registers and their usage.

LSP3 LOAD STACK POINTER Octal Code: 006421



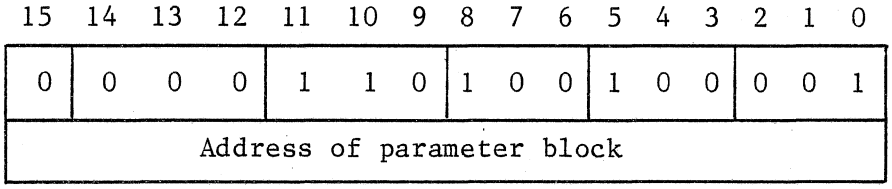
The second word of the instruction is loaded into the coordinate converter stack pointer.

LBOX LOAD VIEWBOX PARAMETERS Octal Code: 006401

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	2-D	3-D No Persp.	3-D Persp.
0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1			
Lv																X	X	X
Bv																X	X	X
Nv																	X	X
Rv																X	X	X
Tv																X	X	X
Fv																	X	X
Xa																		X
Ya																		X
Za																		X

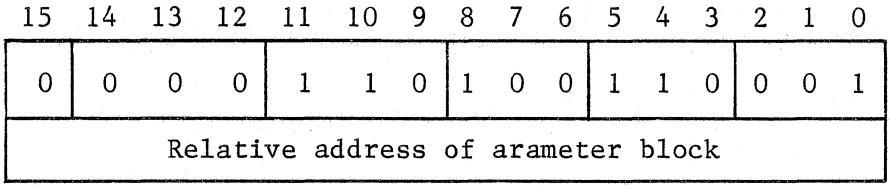
LBOX is used to set the viewbox boundaries (Lv,Bv,Nv,Rv,Tv,Fv) which are used in clipping, and to set the viewing point (Xa,Ya,Za) which is used in generating perspective. The parameter block varies in length from 4 to 9 words as shown above, depending on the currently established dimension (2D/3D, perspective/no perspective).

LBXD LOAD VIEWBOX PARAMETERS DEFERRED Octal Code: 006441



Same as LBOX except that second word of instruction contains pointer to variable length parameter block.

LBDR LOAD VIEWBOX PARAMETERS DEFERRED RELATIVE Octal Code: 006461



Same as LBOX except that second word of instruction is the relative address of variable length parameter block.

LMTX LOAD MATRIX PARAMETERS Octal Code: 006402

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	2D		3D	
0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	0	N-H	H	N-H	H
M_{11}																X	X	X	X
M_{12}																X	X	X	X
M_{13}																	X	X	X
M_{14}																			X
M_{21}																X	X	X	X
M_{22}																X	X	X	X
M_{23}																	X	X	X
M_{24}																			X
M_{31}																X	X	X	X
M_{32}																X	X	X	X
M_{33}																	X	X	X
M_{34}																			X
M_{41}																		X	X
M_{42}																		X	X
M_{43}																		X	X
M_{44}																			X

NOTE

See Section 7.1 for the range of values for elements in the composite matrix.

LMTX is used to load the matrix parameters to be used in the coordinate transformations. The parameter block varies in length from 6 to 16 words, depending on the currently established dimension (2D/3D homogeneous/ non-homogeneous).

The parameters specified occupy locations within a 4 x 4 matrix as shown below:

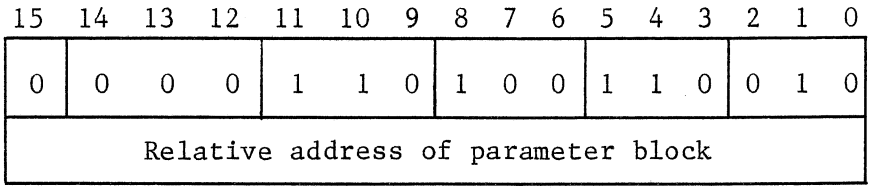
$$\begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix}$$

LMXD LOAD MATRIX PARAMETERS DEFERRED Octal Code: 006442

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	1	0	1	0	1	0
Address of parameter block															

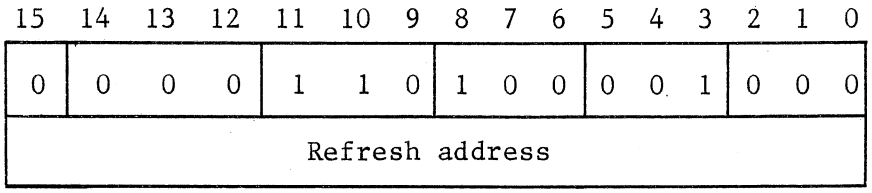
Same as LMTX except that second word of instruction is a pointer to variable length parameter block.

LMDR LOAD MATRIX PARAMETERS DEFERRED RELATIVE Octal Code: 006462



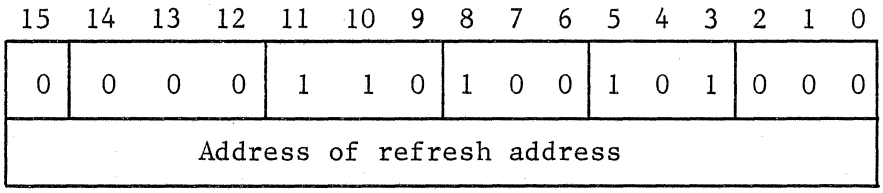
Same as LMTX except that second word of instruction is relative address of variable length parameter block.

LREF LOAD REFRESH ADDRESS Octal Code: 006410



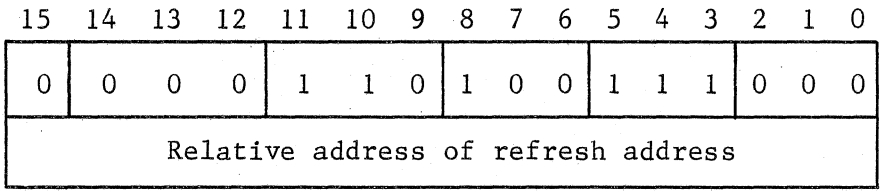
Loads second word of instruction into refresh address register.

LRFD LOAD REFRESH ADDRESS DEFERRED Octal Code: 006450



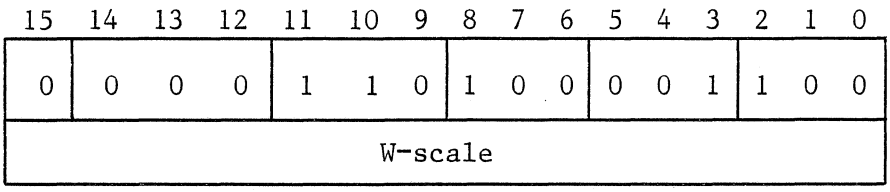
Same as LREF except that second word of instruction is pointer to parameter.

LRDR LOAD REFRESH ADDRESS DEFERRED RELATIVE Octal Code: 006470



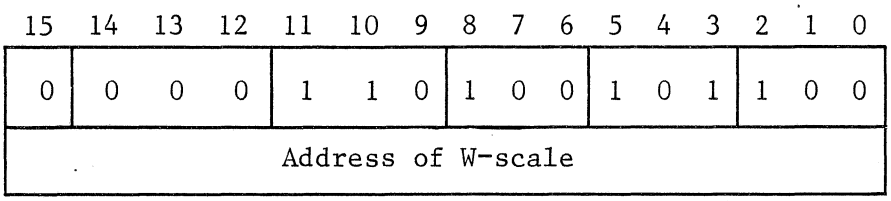
Same as LREF except that second word of instruction is relative address of parameter.

LWSC LOAD W-SCALE Octal Code: 006414



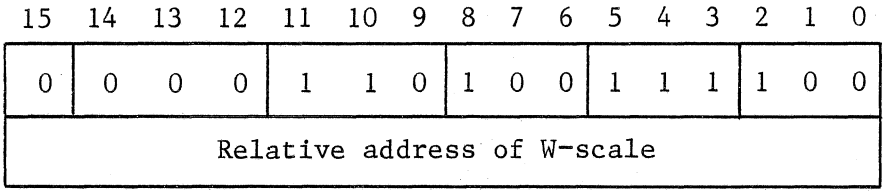
Loads second word of instruction into W-scale register.

LWSD LOAD W-SCALE DEFERRED Octal Code: 006454



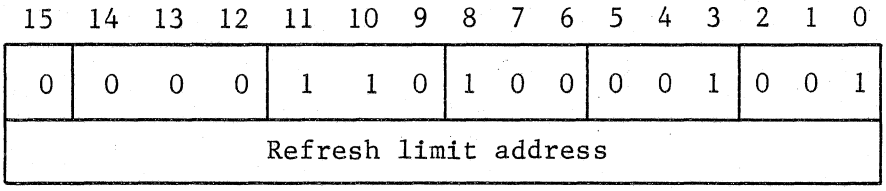
Same as LWSC except that second word of instruction is pointer to parameter.

LWDR LOAD W-SCALE DEFERRED RELATIVE Octal Code: 006474



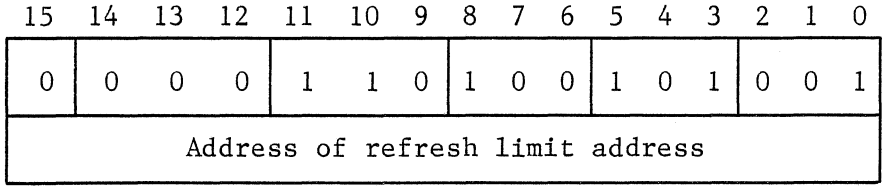
Same as LWSC except that second word of instruction is relative address of parameter.

LLIM LOAD REFRESH LIMIT ADDRESS Octal Code: 006411



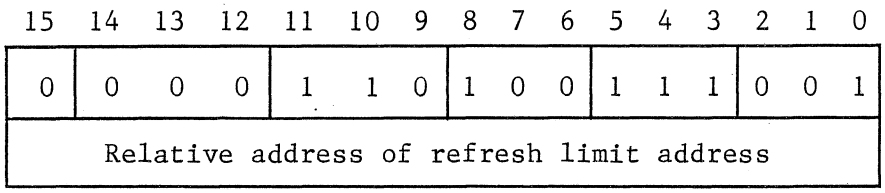
Loads second word of instruction into refresh limit register.

LLMD LOAD REFRESH LIMIT ADDRESS DEFERRED Octal Code: 006451



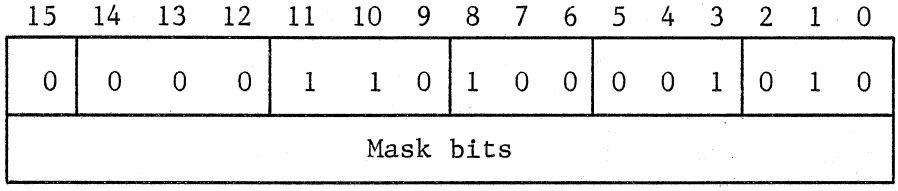
Same as LLIM except that second word of instruction is pointer to parameter.

LLDR LOAD REFRESH LIMIT ADDRESS DEFERRED RELATIVE Octal Code: 006471



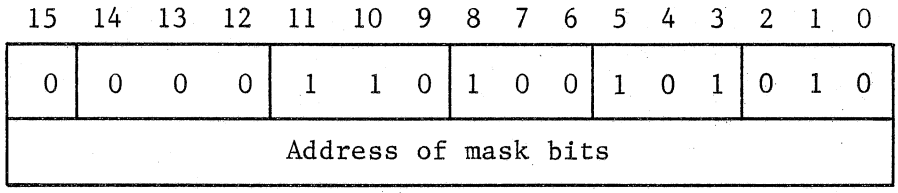
Same as LLIM except that second word of instruction is relative address of parameter.

LMSK LOAD MASK REGISTER Octal Code: 006412



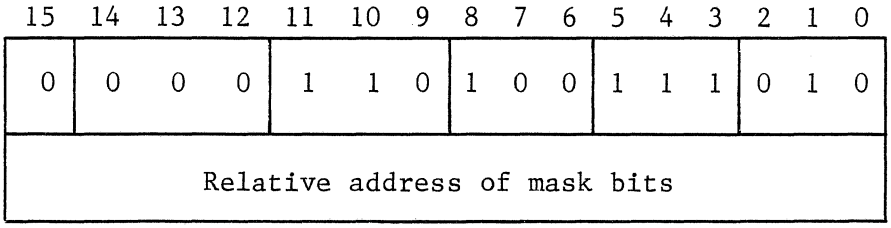
Loads the second word of the instruction into the mask register.

LMKD LOAD MASK REGISTER DEFERRED Octal Code: 006452



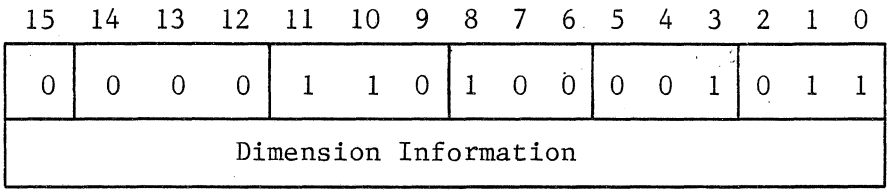
Same as LMSK except that second word of instruction is pointer to parameter.

LKDR LOAD MASK REGISTER DEFERRED RELATIVE Octal code: 006472



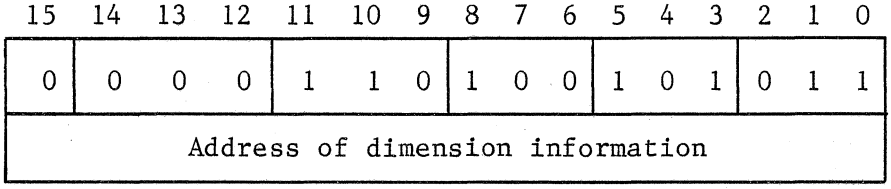
Same as LMSK except that second word of instruction is relative address of parameter.

LDIM LOAD DIMENSION INFORMATION Octal Code: 006413



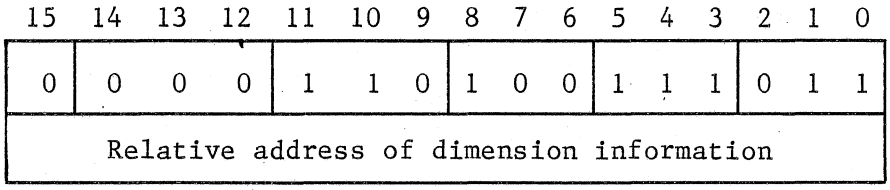
Loads second word of instruction into dimension register.

LDMD LOAD DIMENSION INFORMATION DEFERRED Octal Code: 006453



Same as LDIM except that second word of instruction is pointer to parameter.

LDDR LOAD DIMENSION INFORMATION DEFERRED RELATIVE Octal Code: 006473



Same as LDIM except that second word of instruction is relative address of parameter.

SECTION 4

GRAPHIC CONTROLLER INSTRUCTIONS

Any standard graphic controller instructions can be inserted in line in the image file. These instructions are generally passed directly through the coordinate converter and loaded into the refresh file. Note, however, that no transformations are performed on graphic controller moves and draws. Also, if clipping is enabled, vectors drawn with graphic controller instructions are not clipped. However, graphic controller TEXT and CHAR instructions which attempt to define characters out of the clipping window are clipped.

SECTION 5

COORDINATE CONVERTER REGISTERS

The coordinate converter contains a number of parameter and control registers which can be written to or read from via the display processor.

If the coordinate converter is halted, then these registers can be accessed freely. However, if the coordinate converter is not halted and is executing instructions, accessing a coordinate converter register may cause a bus time-out which will in turn cause an interrupt to the display processor. This occurs because when the coordinate converter is running, it only responds to register accesses for a brief interval before fetching the next instruction. The average time between instruction fetches is several times longer than the interval that defines a bus time-out. Therefore, be certain that the coordinate converter has halted before doing any register accesses.

You can access the stop register (the stop register is the only register that can be accessed at any time) and wait for a stop interrupt to occur. It is then safe to access registers and, if you want, the coordinate converter can be re-started by accessing the continue register.

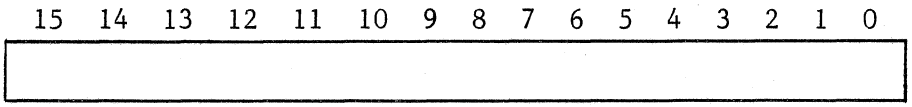
Three of these registers, INZ, CNT, and STOP, are not actually physical registers. However, accessing these registers (either reading or writing) causes the coordinate converter to perform an action. The remainder of the registers are 16 bits long.

REGISTER		MEMORY ADDRESS
MATRIX PARAMETERS REGISTERS	M11	172500
	M12	172502
	M13	172504
	M14	172506
	M21	172510
	M22	172512
	M23	172514
	M24	172516
	M31	172520
	M32	172522
	M33	172524
	M34	172526
	M41	172530
	M42	172532
	M43	172534
	M44	172536
VIEWBOX PARAMETERS REGISTERS	Lv	172540
	Bv	172542
	Nv	172544
	Rv	172546
	Tv	172550
	Fv	172552
PERSPECTIVE PARAMETERS REGISTERS	Xa	172554
	Ya	172556
	Za	172560
W-SCALE REGISTER	WSR	172564
REFRESH ADDRESS REGISTER	RAR	172566
REFRESH LIMIT REGISTER	LIM	172570
BLOCK REGISTER	BLK	172572
DIMENSION REGISTER	DIM	172574
PHOTOPEN STRIKE ADDRESS REGISTER	STR	172600

REGISTER		MEMORY ADDRESS
CC INSTRUCTION REGISTER	CIR	172602
CC PROGRAM COUNTER	CPC	172604
INCOMING POINT REGISTERS	XIN	172606
	YIN	172610
	ZIN	172612
	WIN	172614
MASK REGISTER	MSK	172616
STATUS REGISTER	STAT	172620
STACK POINTER REGISTER	STP	172622
INITIALIZE REGISTER	INZ	172624
CONTINUE REGISTER	CNT	172626
STOP REGISTER	STOP	172630

MATRIX PARAMETER REGISTERS

M11 through M44



The 16 matrix parameter registers contain the elements of the composite matrix used in the coordinate transformation which are generally loaded with a LMAT instruction.

The parameters occupy locations within a 4 x 4 matrix as shown below:

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix}$$

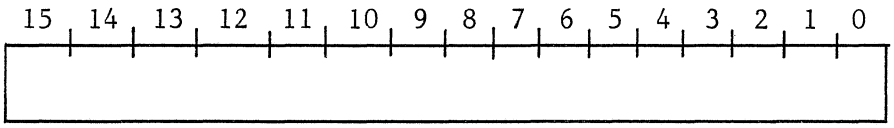
NOTE

See Section 7.1 for the range of values for elements in the composite matrix.

The actual parameters used vary with dimension (2D/3D, homogeneous/non-homogeneous) as shown below:

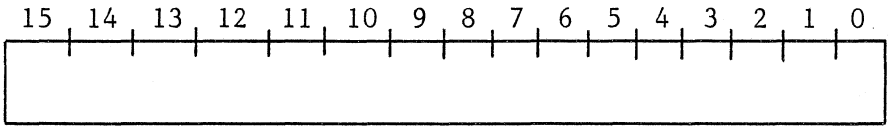
	2D		3D	
	N-H	H	N-H	H
M11	X	X	X	X
M12	X	X	X	X
M13		X	X	X
M14				X
M21	X	X	X	X
M22	X	X	X	X
M23		X	X	X
M24				X
M31	X	X	X	X
M32	X	X	X	X
M33		X	X	X
M34				X
M41			X	X
M42			X	X
M43			X	X
M44				X

Lv LEFT VIEWBOX BOUNDARY REGISTER



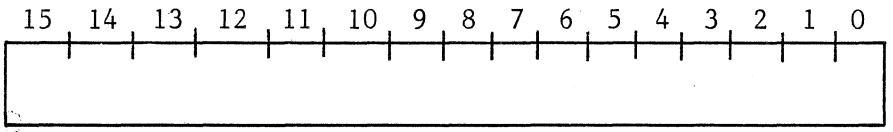
The Lv register contains the minimum X coordinate value which can be displayed with clipping. Its range is ± 32K.

Bv BOTTOM VIEWBOX BOUNDARY REGISTER



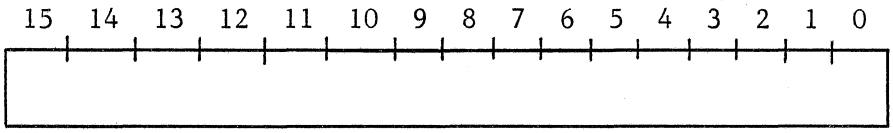
The Bv register contains the minimum Y coordinate value which can be displayed with clipping. Its range is ± 32K.

Nv NEAR VIEWBOX BOUNDARY REGISTER



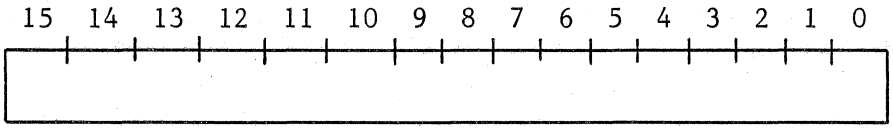
The Nv register contains the minimum Z coordinate value which can be displayed with clipping. Its range is 0 to + 32K.

Rv RIGHT VIEWBOX BOUNDARY REGISTER



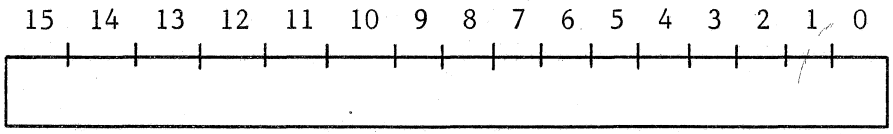
the Rv register contains the maximum X coordinate value which can be displayed with clipping. Its range is ± 32K.

Tv TOP VIEWBOX BOUNDARY REGISTER



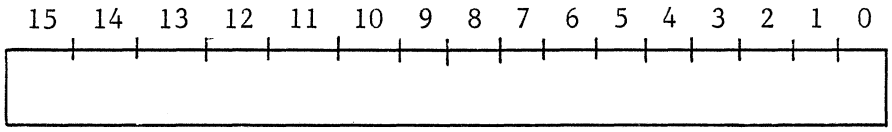
The Tv register contains the maximum Y coordinate value which can be displayed with clipping. Its range is ± 32K.

Fv FAR VIEWBOX BOUNDARY REGISTER



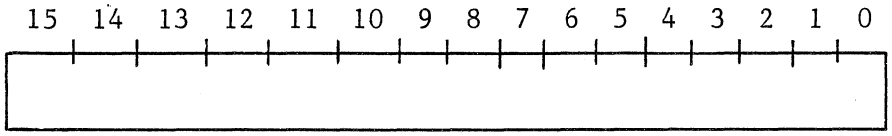
The Fv register contains the maximum Z coordinate value which can be displayed with clipping. Its range is 0 to + 32K.

Xa X PERSPECTIVE PARAMETER REGISTER



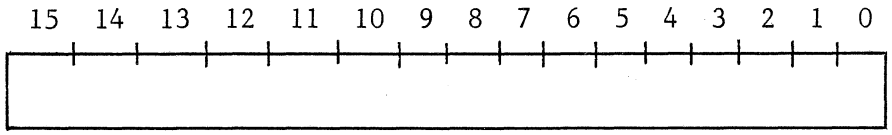
The Xa register contains the X coordinate of the observer's eye position in viewbox space. Its range is ± 32K.

Ya Y PERSPECTIVE PARAMETER REGISTER



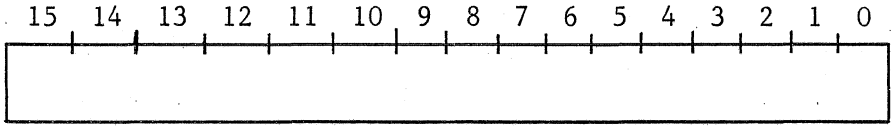
The Ya register contains the Y coordinate of the observer's eye position in viewbox space. Its range is ± 32K.

Za Z PERSPECTIVE PARAMETER REGISTER



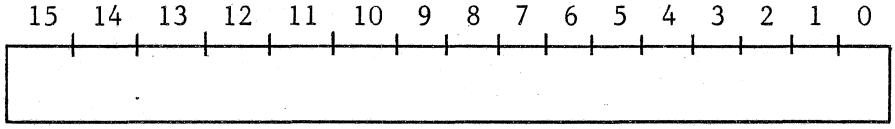
The Za register contains the Z coordinate of the observer's eye position in viewbox space. Its range is 0 to -32K.

WSR W-SCALE REGISTER



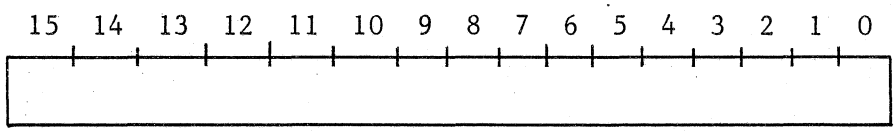
The WSR register contains the exponent value for W, the fourth coordinate. W-scale is used in homogeneous mode to modify the W coordinate by multiplying it by 2 raised to the W-scale power (i.e., W x $2^{W\text{-scale}}$). Its range is $\pm$15.

RAR REFRESH ADDRESS REGISTER



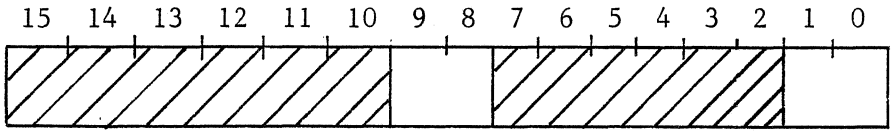
The RAR register contains the 16 low order address bits where the next word of refresh code generated by the coordinate converter is to be stored. The two high order bits are provided by the block register. After each word of refresh code is written into memory, the RAR is incremented by 2.

LIM REFRESH LIMIT REGISTER



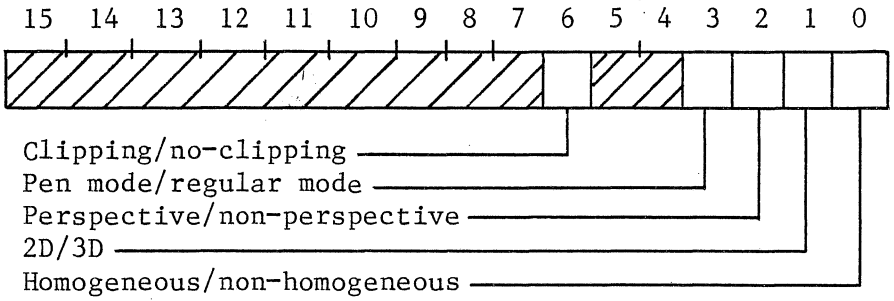
The LIM register contains the 16 low order address bits of the upper limit of memory space allocated for the converted refresh code. The two high order bits are provided by the block register. When an attempt is made by the coordinate converter to place refresh code at this address, a bit is set in the status register indicating an error condition and the coordinate converter halts.

BLK BLOCK REGISTER



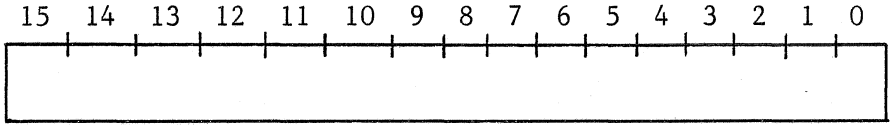
The BLK register contains two sets of the two higher order address bits used by the coordinate converter for memory accesses. Block register bits 1 and 0 are used for all source operations which include accesses to the image file and the coordinate converter stack. Block register bits 9 and 8 are used for all destination operations involving the refresh file.

DIM DIMENSION REGISTER



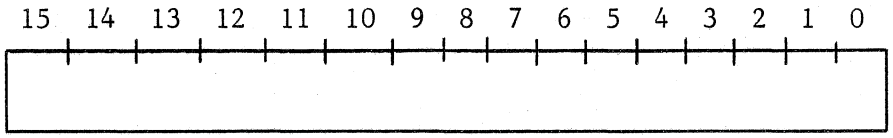
The DIM register contains five bits which establish the manner in which instructions are processed. A '1' in any bit enables the first function in the corresponding pair. A '0' enables its complement. For example, a '1' in bit 6 enables clipping, a '0' enables no-clipping. Each function is discussed in Section 2.

STR PHOTOPEN STRIKE ADDRESS REGISTER



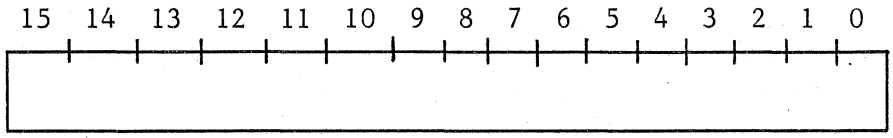
The STR register contains the 16 low order address bits corresponding to a word of refresh code to be used in a pen mode search. Pen mode is discussed in Section 2.

CIR INSTRUCTION REGISTER



The CIR register contains the last instruction which was executed by the coordinate converter.

CPC PROGRAM COUNTER



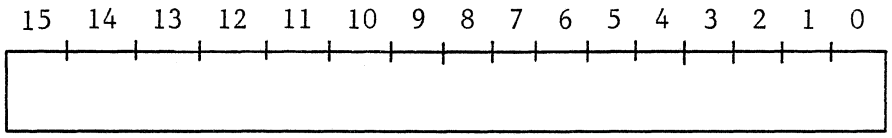
The program counter contains the 16 low order address bits of the next instruction to be executed. As each instruction is fetched from memory, the program counter is automatically incremented by 2. The coordinate conversion process is initiated by writing the starting address of the image file into the program counter.

XIN

YIN INCOMING POINT REGISTERS

ZIN

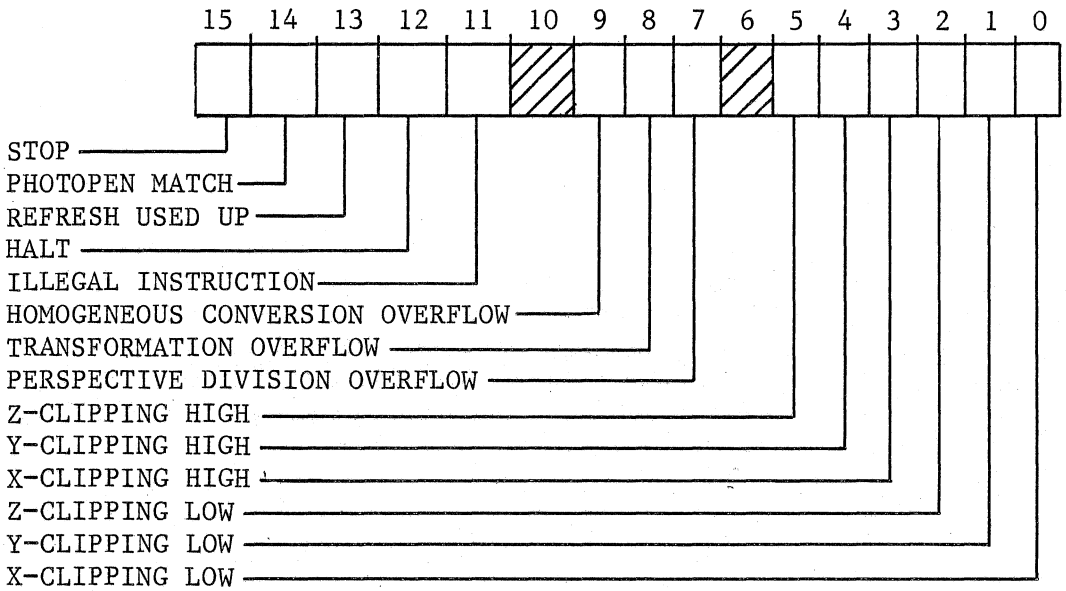
WIN



The incoming point registers contain the coordinates of the last point processed, before transformation. When using absolute moves and draws, the XIN, YIN, and ZIN registers contain the coordinates loaded in from the coordinate block of the last instruction. When using relative moves and draws, the XIN, YIN, and ZIN registers contain the sum of the relative coordinates loaded in from the coordinate block of the last instruction and the resultant coordinates of the previous instruction. When using homogeneous mode, the WIN register contains the homogeneous value loaded in with the last instruction.

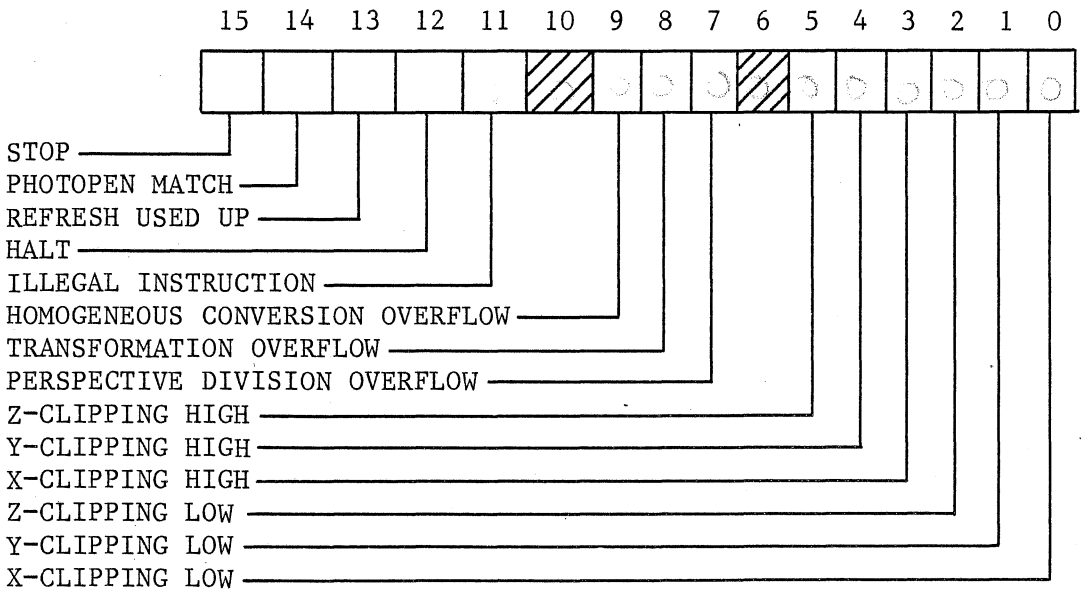
MSK

MASK REGISTER



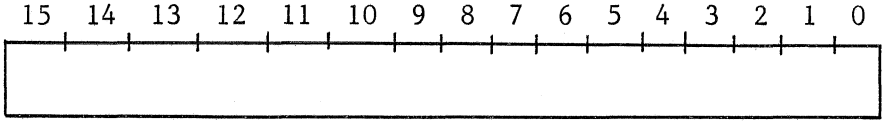
The bits of the mask register are used to allow 14 status conditions to send interrupts to the display processor. A '1' in a mask register bit allows an interrupt if the corresponding bit in the status register goes true ('1'). A '0' in a mask register bit inhibits an interrupt for the corresponding status condition.

STAT STATUS REGISTER



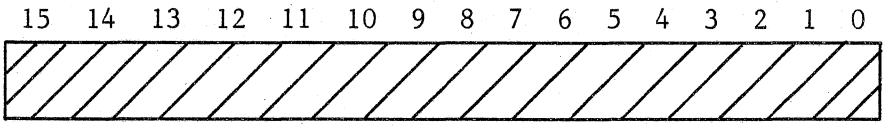
The status register indicates the present state of the coordinate converter. A '1' in any bit indicates that the corresponding condition has gone true and initiates an interrupt to the display processor if the corresponding bit in the mask register is also a '1'. Note that when the status register is read under program control, it is automatically cleared.

STP STACK POINTER



The STP register contains the 16 lower address bits corresponding to the top of the coordinate converter stack. This value is modified when CL3A, CL3R, or RTN3 is executed.

INZ INITIALIZE REGISTER

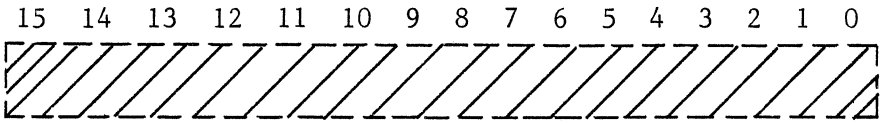


The INZ register is used to initialize the coordinate converter by establishing default values for certain parameter and control registers. Whenever this register is accessed, the following octal values are loaded into the corresponding registers. Note that a GRAPHIC 7® bus reset causes the same initialization.

Mask register (MSK)	177777
Status register (STAT)	0
Block register (BLK)	0
Left viewbox boundary register (Lv)	177000
Bottom viewbox boundary register (Bv)	177000
Near viewbox boundary register (Nv)	0
Right viewbox boundary register (Rv)	777
Top viewbox boundary register (Tv)	777
Far viewbox boundary register (Fv)	40000
Refresh limit register (LIM)	0
W-scale (WSC)	0
Dimension register (DIM)	0

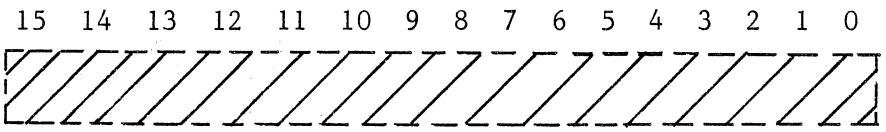
®GRAPHIC 7 is a trademark of Sanders Associates, Inc.

CNT CONTINUE REGISTER



The continue register is used to restart the coordinate converter after it has been stopped by a stop command (see below) or an interrupt such as refresh used up. Accessing this register causes the coordinate converter to resume processing from the point where it was last halted.

STOP STOP REGISTER



The stop register is used to halt the coordinate converter. Accessing this register causes the coordinate converter to complete the instruction in progress, set the stop bit in the status register, and halt. If the stop bit is set in the mask register, an interrupt is generated to the display processor.

SECTION 6

INTERRUPTS

The coordinate converter monitors 14 conditions which, when active, can generate interrupts to the display processor. The current state of these conditions is in the status register ('1' is active). When a condition is active and the corresponding bit in the mask register is also set, then an interrupt signal is automatically sent to the display processor. When the coordinate converter senses that the interrupt has been accepted by the display processor, the active status register bit which initiated the interrupt is cleared, with the exception of data anomaly interrupts.

The 14 conditions are grouped into six interrupt classes (listed below with their corresponding trap address assignments).

Interrupt	Trap Address (octal)
Halt	200
Refresh used up	204
PHOTOPEN match	210
Illegal instruction	214
Data anomaly	220
Stop	224

Halt

When the coordinate converter executes a HLT3 instruction, it sets the halt bit in the status register and halts.

Refresh Used Up

When the coordinate converter attempts to write refresh code to a location which corresponds to the end of the refresh block, it sets the refresh used-up bit in the status register and halts.

PHOTOPEN Match

When the coordinate converter is operating in pen mode and a match occurs between the contents of the PHOTOPEN strike register and the contents of the refresh address register, then it sets the PHOTOPEN match bit in the status register and halts.

Illegal Instruction

When the coordinate converter attempts to execute an illegal instruction, it sets the illegal instruction bit in the status register and then fetches the next instruction.

Data Anomaly

Nine conditions may cause a data anomaly interrupt: three overflow conditions and six clipping conditions. The homogeneous conversion overflow bit in the status register is set in response to an error when de-homogeneizing coordinates. This usually indicates that the resultant coordinates are out of range (±32K).

The transformation overflow bit in the status register is set in response to an error during the transformation process. This generally indicates that the resultant coordinates are out of range (±32K).

The perspective overflow bit in the status register is set in response to an error in the perspective application process. This generally indicates that the coordinate values with perspective applied are out of range (±32K).

The six clipping bits in the status register are individually set when clipping is enabled and a transformed coordinate is found to be outside the respective clipping boundary.

After any of the data anomaly status bits have been set, the coordinate converter completes the instruction in progress and fetches the next one. Note that after a data anomaly interrupt has been accepted by the display processor, the coordinate converter does not clear the status bit which initiated the interrupt. It does, however, ensure that this same condition does not generate a second interrupt. This lets you read the status register to determine which of the nine conditions caused the interrupt.

Stop

When the stop register is accessed by the display processor, the coordinate converter sets the stop bit in the status register and halts.

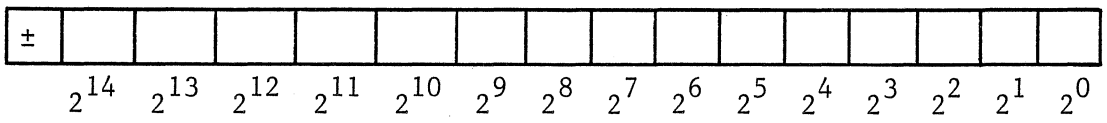
SECTION 7

INSTRUCTION USAGE

7.1 NUMBERING SYSTEM

The 3D Coordinate Converter uses two numbering systems, fixed point integer arithmetic and a modified fractional two's complement arithmetic.

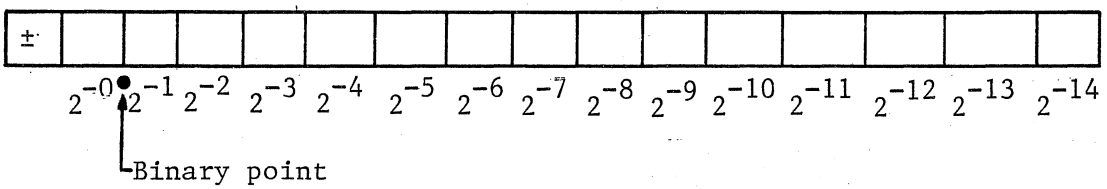
- Fixed Point Integer Arithmetic



The range of values possible is:

$$-32768 \leq \text{number} \leq 32767$$

- Modified Fractional Two's Complement Arithmetic



The range of values possible is:

$$-2.0 \leq \text{number} < 2.0$$

- Use of Applicable Numbering System

  Coordinate values (X,Y,Z) found in 3D instructions are represented in fixed point integer arithmetic. Translation values (Tx, Ty, Tz) found in the composite matrix are also in fixed point integer arithmetic.

  All other numbers, e.g., the 4th coordinate (W-coordinate) found in 3D instructions and all other elements other than Tx, Ty, Tz in the composite matrix are expressed in the modified fractional two's complement arithmetic.

Numbering System Constraints

- Coordinates

  $-32768 \leq X, Y \leq 32767$

  $0 \leq Z \leq 32767$

  $0.0 < W \leq 1.0$

- Composite Matrix (4 x 4) Elements

  Elements

  $-2.0 \leq \{1,1;1,2;1,3;2,1;2,2;2,3;3,1;3,2;3,3\} < 2.0$

  $\{1,4;2,4;3,4\} = 0.0$ (not used)

  $-32768 \leq \{4,1;4,2;4,3\} \leq 32767$ (Tx, Ty, Tz)

  $\{4,4\} = 1.0$

7.2 MATRIX OPERATIONS

In the following discussion, all matrices are shown as 4 x 4 matrices for generality. The demonstrated principles can be applied to other dimensioned matrices as desired. Also, numbers shown in matrices are represented as integers for simplicity: e.g., 1 instead of 40000.

The following functions can be implemented using matrices individually. Any combination of these functions can be implemented by concatenating these matrices into a composite matrix.

- Translation
- Scaling
- Rotation

7.3 TRANSLATION

An object can be translated in X, Y, and/or Z. This may be represented by addition of the offset or translational value T_x, T_y, and/or T_z to the initial coordinate dimensions.

$$X_1 = X_0 + T_x$$
$$Y_1 = Y_0 + T_y$$
$$Z_1 = Z_0 + T_z$$

These equations can be represented in matrix form as:

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & W_1 \end{bmatrix} = \begin{bmatrix} X_0 & Y_0 & Z_0 & W_0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

7.4 SCALING

An object can be scaled in X, Y and/or Z. This can be represented as multiplication of the initial coordinate dimensions by a factor S_x, S_y, and/or S_z.

$$X_1 = S_x X_0$$
$$Y_1 = S_y Y_0$$
$$Z_1 = S_z Z_0$$

These equations can be represented in matrix form as:

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & W_1 \end{bmatrix} = \begin{bmatrix} X_0 & Y_0 & Z_0 & W_0 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7.5 ROTATION

An object can be rotated about any axis in 3-D space through a combination of rotations about the X, Y, and Z axes (using a left-handed coordinate system).

A_x = Angular rotation about X axis (in Y-Z plane)

A_y = Angular rotation about Y axis (in X-Z plane)

A_z = Angular rotation about Z axis (in X-Y plane)

A positive angle is defined to give a counterclockwise rotation when looking towards the origin from the negative axis of rotation. For example, for a positive value of A_x, when looking towards the origin from the negative X axis, the Y-Z plane rotates in a counterclockwise direction.

The following equations represent rotation of a coordinate point X_0, Y_0, Z_0 about the X axis to a point X_1, Y_1, Z_1.

$$\left.\begin{aligned} X_1 &= X_0 \\ Y_1 &= Y_0 \cos A_x - Z_0 \sin A_x \\ Z_1 &= Y_0 \sin A_x + Z_0 \cos A_x \end{aligned}\right\} \quad \text{X rotation}$$

Similarly for rotation about the Y axis:

$$\left.\begin{aligned} X_1 &= X_0 \cos A_y + Z_0 \sin A_y \\ Y_1 &= Y_0 \\ Z_1 &= -X_0 \sin A_y + Z_0 \cos A_y \end{aligned}\right\} \quad \text{Y rotation}$$

And for rotation about the Z axis:

$$\left.\begin{aligned} X_1 &= X_0 \cos A_z - Y_o \sin A_z \\ Y_1 &= X_0 \sin A_z + Y_0 \cos A_z \\ Z_1 &= Z_0 \end{aligned}\right\} \quad Z \text{ rotation}$$

These equations are represented in matrix form as:

For X rotation:

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & W_1 \end{bmatrix} = \begin{bmatrix} X_0 & Y_0 & Z_0 & W_0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos A_x & \sin A_x & 0 \\ 0 & -\sin A_x & \cos A_x & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

For Y rotation:

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & W_1 \end{bmatrix} = \begin{bmatrix} X_0 & Y_0 & Z_0 & W_0 \end{bmatrix} \begin{bmatrix} \cos A_y & 0 & -\sin A_y & 0 \\ 0 & 1 & 0 & 0 \\ \sin A_y & 0 & \cos A_y & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

For Z rotation:

$$[X1\ Y1\ Z1\ W1] = [X0\ Y0\ Z0\ W0] \begin{bmatrix} \cos Az & \sin Az & 0 & 0 \\ -\sin Az & \cos Az & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7.6 MATRIX CONCATENATION

To combine more than one function of scaling, translation, or rotation, it is necessary to concatenate the individual matrices involved, preparing one composite matrix for transmission to the coordinate converter. Matrices which are to be multiplied together must have the same dimensions: 3 x 3 for 2D; 4 x 4 for 3D. The multiplication process is illustrated below for a 3 x 3 case.

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \quad \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} \quad \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}$$

where
$$C_{11} = A_{11}B_{11} + A_{12}B_{21} + A_{13}B_{31}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22} + A_{13}B_{32}$$
$$C_{13} = A_{11}B_{13} + A_{12}B_{23} + A_{13}B_{33}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21} + A_{23}B_{31}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22} + A_{23}B_{32}$$
$$C_{23} = A_{21}B_{13} + A_{22}B_{23} + A_{23}B_{33}$$
$$C_{31} = A_{31}B_{11} + A_{32}B_{21} + A_{33}B_{31}$$
$$C_{32} = A_{31}B_{12} + A_{32}B_{22} + A_{33}B_{32}$$
$$C_{33} = A_{31}B_{13} + A_{32}B_{23} + A_{33}B_{33}$$

If you want to rotate about more than one coordinate axis, it is necessary to concatenate the rotational matrices.

Let R_x = X axis rotational matrix

R_y = Y axis rotational matrix

R_z = Z axis rotational matrix

Then $R_c = R_x R_y R_z$

Where R_c represents the composite rotational matrix.

NOTE

This matrix provides rotation of an object with respect to the origin. If you want to rotate an object about the center of the object, and the center of the object does correspond to the origin, then it is necessary to translate the object to the origin, rotate it, and then translate it back to its initial position.

Let X_c, Y_c, and Z_c represent the coordinates of the center of the object to be rotated.

Then $T_{r1} =$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -X_c & -Y_c & -Z_c & 1 \end{bmatrix}$$

This is the matrix that translates the object to the origin.

$$T_{r2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ X_c & Y_c & Z_c & 1 \end{bmatrix}$$

This is the matrix that retranslates the object back to its initial position. So, the composite rotational matrix for this case is:

$$R_c = T_{r1} R_x R_y R_z T_{r2}$$

If you then want to scale this object and to translate it to a given location, two more concatenations are necessary.

Let S = scaling matrix
T = translational matrix

Then $C = TSR_c$

Where C represents the final composite matrix to be passed to the coordinate converter.

7.7 SAMPLE IMAGE FILE FOR CONVERTING A CUBE

```
                                        ;
                                        ;       SET UP PARAMETERS
                                        ;
002000                                          LDIM    105             ;SET CLIPPING,3D,PERSPECTIVE,AND
002000  006413  000105
                                                                        ;HOMOGENEOUS MODE IN DIMENSION REGISTER
002004                                          LMXD    IMAT            ;LOAD DEFERRED MATRIX FROM IMAT
002004  006442  002134
002010                                          LBXD    BOX             ;LOAD DEFERRED VIEWBOX FROM BOX
002010  006441  002174
002014                                          LREF    20000           ;REFRESH CODE TO BE GENERATED
002014  006410  020000
                                                                        ;AT LOCATION 20000
002020                                          LLIM    20100           ;REFRESH LIMIT AT 20100
002020  006411  020100
002024                                          LMSK    30000           ;ENABLE HALT AND REFRESH USED UP INTERRUPT
002024  006412  030000
                                        ;
                                        ;
                                        ;       DEFINE CUBE
                                        ;
                                        ;       DRAW FRONT SQUARE
                                        ;
002030                                          AMD3    PT1             ;MOVE TO LOWER LEFT
002030  006444  002216
002034                                          ADD3    PT2             ;DRAW TO UPPER LEFT
002034  006446  002226
002040                                          ADD3    PT3             ;TO UPPER RIGHT
002040  006446  002236
002044                                          ADD3    PT4             ;TO LOWER RIGHT
002044  006446  002246
002050                                          ADD3    PT1             ;BACK TO LOWER LEFT
002050  006446  002216
                                        ;
                                        ;       DRAW REAR SQUARE
                                        ;
002054                                          ADD3    PT5             ;DRAW STRAIGHT BACK
002054  006446  002256
002060                                          ADD3    PT6             ;TO UPPER LEFT
002060  006446  002266
002064                                          ADD3    PT7             ;TO UPPER RIGHT
002064  006446  002276
002070                                          ADD3    PT8             ;TO LOWER RIGHT
002070  006446  002306
002074                                          ADD3    PT5             ;BACK TO LOWER LEFT
002074  006446  002256
                                        ;
                                        ;       CONNECT FRONT TO BACK
                                        ;
002100                                          AMD3    PT2             ;MOVE TO UPPER LEFT
002100  006444  002226
002104                                          ADD3    PT6             ;DRAW STRAIGHT BACK


 DRASS.616   22-JUN-79  15:29  PAGE 1-1
SRC

002104  006446  002266
002110                                          AMD3    PT4             ;MOVE TO LOWER RIGHT
002110  006444  002246
002114                                          ADD3    PT8             ;DRAW STRAIGHT BACK
002114  006446  002306
002120                                          AMD3    PT3             ;MOVE TO UPPER RIGHT
002120  006444  002236
002124                                          ADD3    PT7             ;DRAW STRAIGHT BACK
002124  006446  002276
                                        ;
002130                                          RTRN                    ;GRAPHIC CONTROLLER RETURN
002130  002300
002132                                          HLT3                    ;HALT - END OF IMAGE FILE
002132  006422
                                        ;
                                        ;
                                        ;       COMPOSITE IMAGE MATRIX
                                        ;
002134  040000  000000  000000  IMAT:   .WORD   UNITY,0,0,0      ;IDENTITY MATRIX - 4X4
002142  000000
002144  000000  040000  000000          .WORD   0,UNITY,0,0
002152  000000
002154  000000  000000  040000          .WORD   0,0,UNITY,0
002162  000000
002164  000000  000000  000000          .WORD   0,0,0,UNITY
002172  040000
                                        ;
                                        ;       VIEWBOX
                                        ;
002174  176400  176400  000000  BOX:    .WORD   -1400,-1400,0   ;LV,BV,NV
002202  001400  001400  040000          .WORD   1400,1400,40000 ;RV,TV,FV
002210  000000  000000  170000          .WORD   0,0,-10000      ;XA,YA,ZA
                                        ;
                                        ;       COORDINATE POINTS
                                        ;
002216  177400  177400  001000  PT1:    .WORD   -400,-400,1000,40000
002224  040000
002226  177400  000400  001000  PT2:    .WORD   -400,400,1000,40000
002234  040000
002236  000400  000400  001000  PT3:    .WORD   400,400,1000,40000
002244  040000
002246  000400  177400  001000  PT4:    .WORD   400,-400,1000,40000
002254  040000
002256  177400  177400  002000  PT5:    .WORD   -400,-400,2000,40000
002264  040000
002266  177400  000400  002000  PT6:    .WORD   -400,400,2000,40000
002274  040000
002276  000400  000400  002000  PT7:    .WORD   400,400,2000,40000
002304  040000
002306  000400  177400  002000  PT8:    .WORD   400,-400,2000,40000
002314  040000
```

SECTION 8

ASSOCIATED GCP+ INSTRUCTIONS

The Coordinate Converter may be programmed through the use of GCP+ instructions or through the use of the FSP subroutine support package. Both methods are described below.

8.1 PROGRAMMING THE 2-D/3-D COORDINATE CONVERTER IN GCP

By using the register update (RU) and give register (GR) commands, the GCP+ programmer may read and write all registers associated with the 2-D/3-D coordinate converter.

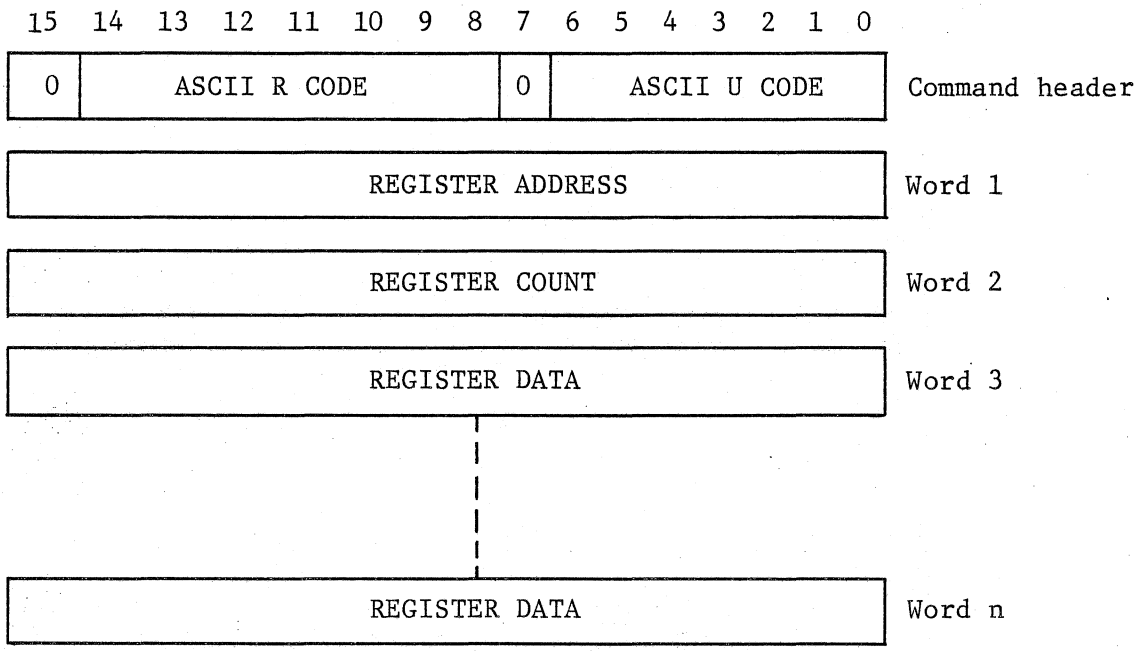
This allows complete host control to perform such functions as:

- Set matrix parameters
- Set viewbox parameters
- Set perspective parameters
- Set various control parameters
  - Scale select
  - Refresh limits select
  - Source/destination of conversion process
  - Homogeneous/non-homogeneous select
  - 2-D/3-D select
  - Perspective/no perspective select
- Start 2-D/3-D coordinate converter
- Activate 2-D/3-D coordinate converter for a PHOTOPEN search
- Selectively establish the desired interrupt control

When 2-D/3-D coordinate converter interrupts are generated, an appropriate TS message is returned to the host computer.

NOTE

Refer to Sanders publication H-79-0348 for more information on GCP+.

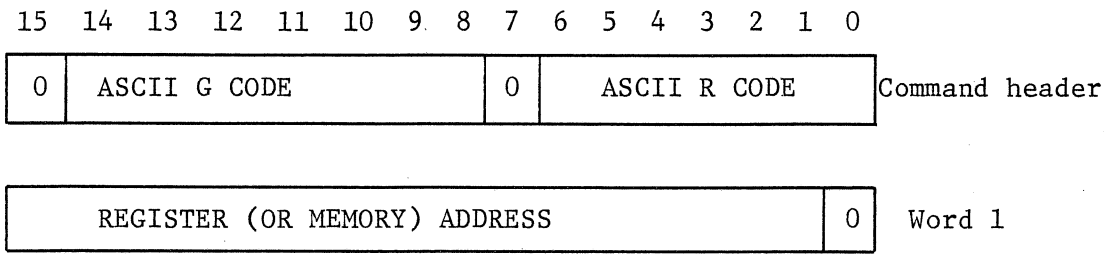


The RU message is a variable length message that is used to update a series of registers in the I/O address of the hardware. Word 1 contains the address of the first register to be updated. Valid register addresses are in the range of 160000-177777 (octal). Word 2 contains the register count indicating the number of successive registers to be updated. Words 3 through n contain the data values to be loaded into each register.

NOTE

The RU message does not change the current memory bank selection. It is also possible to interpret register address as memory address in the RU message. When updating memory address, the user must take into account memory mapping. Memory addresses in the range of 020000 to 077777 are subject to memory mapping.

GR (H-G7) GIVE REGISTER Command header code (octal): 043522



The GR message is a two-word message used by the host computer to obtain the contents of the GRAPHIC 7 register specified by the register address in word 1. The contents of any register having an assigned address may be obtained in this manner. If required, GCP+ automatically halts the graphic controller before the data is obtained and then restarts it at the completion of the operation. In response to a GR message, GCP+ sends an RR (return register) message to the host.

Although the intent of the GR message is to permit the contents of the registers to be read, it can also be used to read the contents of GRAPHIC 7 memory address. When it is used to read a memory address, the address specified in word 1 must be that of an even-numbered byte. If the address of an odd-numbered byte is specified, GCP+ causes an XX (error status) message to be sent to the host.

NOTE

When the GR message is used on a large memory system, the following restrictions must be taken into account:

1. Addresses in the range of 000000-017776 are directly addressable.

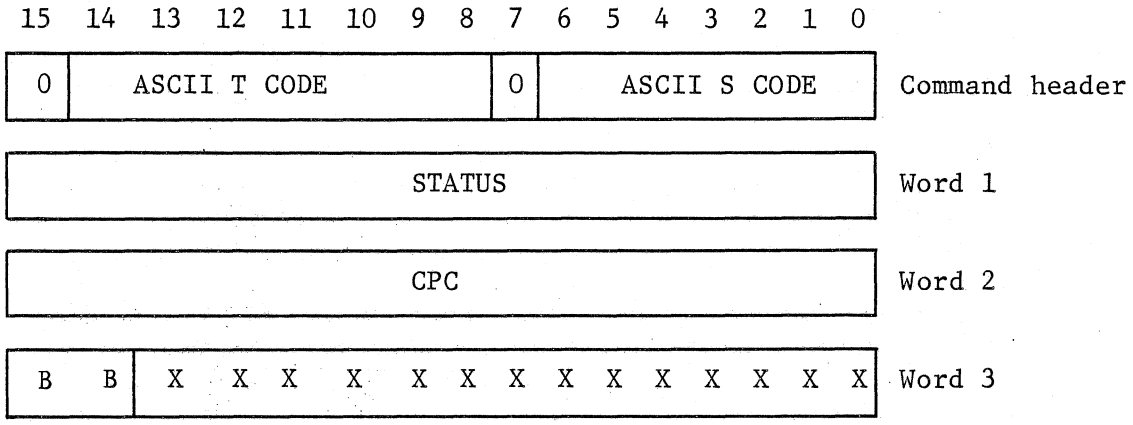
2. Addresses in the range of 020000-077776 are subject to memory mapping.

3. Addresses in the range of 100000-177776 are directly addressable.

4. Addresses in the range of 120000-177776 are related to ROM and I/O device registers.

TS (G7-H) 2-D/3-D COORDINATE CONVERTER STATUS

Command header code (octal): 052123



The 2-D/3-D coordinate converter can generate 14 interrupt conditions, provided that the corresponding mask bits are enabled. The TS message is returned to the host computer when a 2-D/3-D coordinate converter interrupt condition occurs.

Word 1 contains the contents of the 2-D/3-D coordinate converter status register. Each bit in this register corresponds to an interrupt condition. One or more of these bits sets to indicate the type of interrupt condition detected.

Word 2 is the value of the 2-D/3-D coordinate converter program counter.

Word 3 contains the 2-D/3-D coordinate converter block register bits used for source operations at the time of the interrupt condition.

Bits 15 14	Bank Number
0 0	0
0 1	1
1 0	2
1 1	3

8.2 FSP 3D COORDINATE CONVERTER PROGRAMMING

This section describes the FSP support for the 3D Coordinate Converter.

NOTE

FSP (Model 7764) is a collection of Fortran callable routines residing in the host.

This FSP support is included in the FSP subroutine package and is provided at no additional charge. The following ten subroutines make up this option:

INIT3D	Initialize 3D system
SCAL3D (ZL,ZU)	Define Z coordinate system
CCBLK	Create 3D coordinate converter block
MOVE3D (X,Y,Z,MODE)	Create 3D move graphic order
DRAW3D (X,Y,Z,MODE)	Create 3D draw graphic order
T3D2D (IPAG3D,IPAG2D)	Transform 3D page to 2D page
MTRX3D (ARRAY)	Update composite matrix in CCBLK
VIEWPT (X,Y,Z)	Update view point in CCBLK
VIEWBX (LV,RV,TV,BV,NV,FV)	Update view box in CCBLK

The remaining pages describe each subroutine in detail.

8.2.1 INIT3D - Initialize 3D

NAME: INIT3D

FUNCTION: Initialize FSP variables for 3D Coordinate Converter support.

CALLING FORMAT: CALL INIT3D

DESCRIPTION OF PARAMETERS: None

DETAILED DESCRIPTION:

This routine sets default values for the Z-axis user coordinates. The default lower boundary is 0 and the default higher boundary is 32767.

8.2.2 SCAL3D - Define Z Coordinate System

NAME: SCAL3D

FUNCTION: Set the user coordinate values for the Z-axis user coordinates (third coordinate).

CALLING FORMAT: CALL SCAL3D (ZL,ZU)

DESCRIPTION OF PARAMETERS:

ZL = Real variable supplied by the user which specifies the value to be assigned to the lower boundary of the Z-axis in the user coordinate system. Note that ZL is coincident with the screen surface.

ZU = Real variable supplied by the user which specifies the value to be assigned to the upper boundary of the Z-axis in the user coordinate system.

DETAILED DESCRIPTION:

This routine sets the Z-axis user coordinates (third coordinate) to the values passed (ZL and ZU). This allows the caller to define the Z-axis near and far coordinates in real numbers. The 3D move and draw subroutines convert a coordinate in real numbers to an integer display coordinate. This conversion process is based upon the values of ZL and ZU. Without a call to SCAL3D, the Z-axis of the user coordinate system is equal to the default Z-axis coordinates, i.e., ZL=0 and ZU=32767. Note that the Z-axis is defined within a left-handed coordinate system. ZL is the Z-axis point that corresponds to the screen and ZU is the Z-axis point that is the furthest from the screen extending into the screen. The value of ZU must be greater than the value of ZL or unpredictable results will occur.

8.2.3 CCBLK - Initialize Viewbox, Viewpoint, and Matrix

NAME: CCBLK

FUNCTION: Generate Coordinate Converter instructions to initialize the viewbox boundaries, the viewpoint and the matrix parameters.

CALLING FORMAT: CALL CCBLK

DESCRIPTION OF PARAMETERS: None

DETAILED DESCRIPTION:

This routine generates two Coordinate Converter instructions and places them at the beginning of the currently opened page. The first instruction generated (LBOX) initializes the viewbox boundaries, which are used in clipping, and the viewpoint which is used in generating perspective. The viewbox and viewpoint parameters are initialized as follows:

Parameter	Value Set to by CCBLK
Viewbox left (minimum X)	-512.
Viewbox bottom (minimum Y)	-512.

Parameter	Value Set to by CCBLK
Viewbox near (minimum Z)	0.
Viewbox right (maximum X)	511.
Viewbox top (maximum Y)	511.
Viewbox far (maximum Z)	32767.
X Viewpoint	0.
Y Viewpoint	0.
Z Viewpoint	-32767.

The second instruction generated (LMTX) initializes the matrix parameters which are used in the coordinate transformation process. All matrix elements except the scale factors are initialized to 0. The scale factors, matrix elements (1,1), (2,2), and (3,3) are set to 256. This is equivalent to 1/64 in the fractional two's complement notation (see section 7.1) and is the default scale factor. The combination of the LBOX and LMTX instructions at the beginning of a page is referred to as the CCBLK of the page.

8.2.4 MOVE3D - Create 3D Move Graphic Order

NAME: MOVE3D

FUNCTION: Generates either an absolute or relative 3D move graphic order and places it at the mark position of the currently opened page.

CALLING FORMAT: CALL MOVE3D (X,Y,Z,MODE)

DESCRIPTION OF PARAMETERS:

X,Y,Z = Real variables supplied by the user which specify the 3D coordinate of the desired beam position. The coordinate is specified in the user coordinate system.

MODE = An integer variable supplied by the caller which identifies the type of graphic instruction to be generated. When MODE = 0 an absolute MOVE is implied and when MODE=1 a relative MOVE is implied. When MODE=0, the coordinate (X,Y,Z) specifies an absolute 3D coordinate. When MODE=1, the coordinate (X,Y,Z) specifies an offset to be moved from the current beam position.

DETAILED DESCRIPTION:

This routine converts the coordinate values specified to absolute screen coordinates and generates either an absolute or relative 3D move graphic order. This graphic order is placed at the mark position of the currently opened page. Note that relative moves are restricted to 1/2 of the screen size.

8.2.5 DRAW3D - Create 3D Draw Graphic Order

NAME: DRAW3D

FUNCTION: Generates either an absolute or relative 3D draw graphic order and places it at the mark position of the currently opened page.

CALLING FORMAT: CALL DRAW3D (X,Y,Z,MODE)

DESCRIPTION OF PARAMETERS:

X,Y,Z = Real variables supplied by the user which specify the 3D coordinate of the end point of a line to be drawn. The coordinate is in the user coordinate system.

MODE = An integer variable supplied by the caller which identifies the type of graphic instruction to be generated. When MODE=0, an absolute DRAW is implied and when MODE=1, a relative DRAW is implied. When MODE=0, the coordinate (X,Y,Z) specifies the absolute coordinate of the end point of a line to be drawn. When MODE=1, the coordinate specifies the offsets to be used in drawing a line from the current beam position to a new position.

DETAILED DESCRIPTION:

This routine converts the coordinate values specified to absolute screen coordinates and generates either an absolute or relative 3D draw graphic order. This graphic order is placed at the mark position of the currently opened page. Note that relative draws are restricted to 1/2 of the screen size.

8.2.6 T3D2D - Transform 3D to 2D

NAME: T3D2D

FUNCTION: Transform a page of graphic orders into a page which consists entirely of 2D graphic orders.

CALLING FORMAT: CALL T3D2D (IPAG3D,IPAG2D)

DESCRIPTION OF PARAMETERS:

IPAG3D = An integer variable supplied by the user which specifies the number of the 3D page which is to be transformed. IPAG3D must be in the range:

$1 < IPAG3D < 256, IPAGE3D \neq IPAG2D$

IPAG2D = An integer variable supplied by the user which specifies the number of the 2D page in which the transformed graphic orders are to be placed. IPAG2D must be in the range:

$1 < IPAG2D < 256, IPAG2D \neq IPAG3D$

DETAILED DESCRIPTION:

This routine sets up the coordinate converter to perform a 3D to 2D transformation on the page of graphic instructions specified by IPAG3D. The Coordinate Converter is started and a block of transformed graphic instructions is output to the page specified by IPAG2D. Note that unpredictable results will occur if the 2D output file is not large enough to accommodate the transformed graphic instructions. Refer to Appendix D of the Graphic 7 Fortran Support Package (FSP) User's Manual to determine the necessary output file size.

8.2.7 MTRX3D - Compute and Replace Matrix Parameters

NAME: MTRX3D

FUNCTION: Compute matrix parameters and generate coordinate converter instruction to update the matrix parameters.

CALLING FORMAT: CALL MTRX3D (ARRAY)

DESCRIPTION OF PARAMETERS:

ARRAY = A 12 element real array supplied by the user which specifies the scaling, translation, and rotation factors necessary for computing the matrix parameters. The array parameters and their valid ranges are specified below:

ARRAY Element	Definition	Range
1	X-Pre Translation	$-(XU-XL) \leq XPRE \leq +(XU-XL)$
2	Y-Pre Translation	$-(YU-YL) \leq YPRE \leq +(YU-YL)$
3	Z-Pre Translation	$-(ZU-ZL) \leq ZPRE \leq +(ZU-ZL)$
4	X Scale Factor	$1/256 \leq XSC < 128$

ARRAY Element	Definition	Range
5	Y Scale Factor	$1/256 \leq YSC < 128$
6	Z Scale Factor	$1/256 \leq ZSC < 128$
7	X Rotation (Y-Z Plane)	any real number (in radians)
8	Y Rotation (X-Z Plane)	any real number (in radians)
9	Z Rotation (X-Y Plane)	any real number (in radians)
10	X-Post Translation	$-(XU-XL) \leq XPOST \leq +(XU-XL)$
11	Y-Post Translation	$-(YU-YL) \leq YPOST \leq +(YU-YL)$
12	Z-Post Translation	$-(ZU-ZL) \leq ZPOST \leq +(ZU-ZL)$

The variables used above are defined in the SCALE and SCAL3D subroutine descriptions.

e.g. CALL SCALE (XL,YL,XU,YU)

CALL SCAL3D (ZL,ZU)

DETAILED DESCRIPTION:

This routine uses the array parameters passed to compute new matrix parameters. These parameters are computed as indicated in Appendix A and are entered into the CCBLK of the currently opened page, replacing the previous matrix parameters. This routine must be called each time the user wishes to change translation, scaling, or rotation factors.

8.2.8 VIEWPT - Update View Point in CCBLK

NAME: VIEWPT

FUNCTION: Updates view point parameters in the CCBLK of the currently opened 3D page.

CALLING FORMAT: CALL VIEWPT (X,Y,Z)

DESCRIPTION OF PARAMETERS:

(X,Y,Z) = Real variables supplied by the user which specify the viewing point which is to be used in generating perspective. These parameters are specified in user coordinates. The valid ranges for these parameters are:

$$|X| \leq \frac{XU-XL}{2}$$

$$|Y| \leq \frac{YU-YL}{2}$$

$$0 < Z \leq ZU-ZL$$

DETAILED DESCRIPTION:

This routine updates the view point in the currently opened page. The CCBLK view point parameters, generated by a previous call to the CCBLK routine, are updated. This routine must be called in order to change the view point. Note that the view point parameter range limits specified above are constrained by the host computer word size.

8.2.9 VIEWBX - Update Viewbox

NAME: VIEWBX

FUNCTION: Update view box boundaries in the currently opened page.

CALLING FORMAT: CALL VIEWBX (LV,RV,BV,TV,NV,FV)

DESCRIPTION OF PARAMETERS:

LV = Real variable supplied by the user which specifies the viewbox left boundary (minimum X). The valid range for LV is: XL < LV < RV

RV = Real variable supplied by the user which specifies the viewbox right boundary (maximum X). The valid range for RV is: LV < RV < XU

BV = Real variable supplied by the user which specifies the viewbox bottom boundary (minimum Y). The valid range for BV is: YL < BV < TV

TV = Real variable supplied by the user which specifies the viewbox top boundary (maximum Y). The valid range for TV is: BV < TV < YU

NV = Real variable supplied by the user which specifies the viewbox near boundary (minimum Z). The valid range for NV is: ZL $\leq$ NV < FV

FV = Real variable supplied by the user which specifies the viewbox far boundary (maximum Z). The valid range for FV is: NV < FV $\leq$ ZU

The parameters LV, RV, BV, TV, NV and FV are all specified in the user coordinate system.

DETAILED DESCRIPTION:

This routine updates the viewbox boundaries in the currently opened page. The CCBLK viewbox parameters, generated by a previous call to the CCBLK routine, are updated. This routine must be called in order to change the viewbox.

8.3 FSP 2D COORDINATE CONVERTER PROGRAMMING

This section describes the FSP support for the 2D Coordinate Converter.

NOTE

FSP (Model 7764) is a collection of Fortran callable routines residing in the host.

This FSP support is included in the FSP subroutine package and is provided at no additional charge. The following six subroutines make up this option:

CC2DBL	Create 2D converter block
MOVE2D (X, Y, MODE)	Create 2D move graphic order
DRAW2D (X, Y, MODE)	Create 2D draw graphic order
T2D2D (IGRAPH, IPAG2D)	Transform graphic page to 2D page
MTRX2D (ARRAY)	Update 2D composite matrix in CC2DBL
VI2DBX (LV, RV, BV, TV)	Update view box in CC2DBL

The remaining pages describe each subroutine in detail.

8.3.1 CC2DBL - Initialize 2D Viewbox, 2D Viewpoint, and 2D Matrix

NAME: CC2DBL

FUNCTION: Generate Coordinate Converter instructions to initialize the viewbox boundaries and matrix parameters.

CALLING FORMAT: CALL CC2DBL

DESCRIPTION OF PARAMETERS: NONE

DETAILED DESCRIPTION:

This routine generates two Coordinate Converter instructions and places them at the beginning of the currently opened page. The first instruction generated (LBOX) initializes the viewbox boundaries, which are used in clipping. The viewbox parameters are initialized as follows:

Parameter	Value Set to by CC2DBL
Viewbox left (Minimum X)	-512.
Viewbox bottom (Minimum Y)	-512.
Viewbox right (Maximum X)	511.
Viewbox top (Maximum Y)	511.

The second instruction generated (LMTX) initializes the matrix parameters which are used in the transformation process. All matrix

elements except the scale factors are initialized to 0. The scale factors, matrix elements (1,1) and (2,2), are set to 256. This is equivalent to 1/64 in the fractional two's complement notation (see section 7.1) and is the default scale factor. The combination of the LBOX and LMTX instructions at the beginning of a page is referred to as the CC2DBL of the page.

8.3.2 MOVE2D - Create 2D Move Graphic Order

NAME: MOVE2D

FUNCTION: Generates either an absolute or relative 2D move graphic order and places it at the mark position of the currently opened page.

CALLING FORMAT: CALL MOVE2D (X, Y, MODE)

DESCRIPTION OF PARAMETERS:

X, Y = Real variables supplied by the user which specify the 2D coordinate of the desired beam position. The coordinate is specified in the user coordinate system.

MODE = An integer variable supplied by the caller which identifies the type of graphic instruction to be generated. When MODE=0, an absolute MOVE is implied. When MODE=1, a relative MOVE is implied. When MODE=0, The coordinate (X, Y) specifies an absolute 2D coordinate. When MODE=1, the coordinate (X, Y) specifies an offset to be moved from the current beam position.

DETAILED DESCRIPTION:

This routine converts the coordinate values specified to absolute screen coordinates and generates either an absolute or relative 2D move graphic order. This graphic order is placed at the mark position of the currently opened page. Note that relative moves are restricted to 1/2 of the screen size.

8.3.3 DRAW2D - Create 2D Draw Graphic Order

NAME: DRAW2D

FUNCTION: Generates either an absolute or relative 2D draw graphic order and places it at the mark position of the currently opened page.

CALLING FORMAT: CALL DRAW2D (X, Y MODE)

DESCRIPTION OF PARAMETERS:

X, Y = Real variables supplied by the user which specify the 2D coordinate of the end point of a line to be drawn. The coordinate is in the user coordinate system.

MODE = An integer variable supplied by the caller which identifies the type of graphic instruction to be generated. When MODE=0, a absolute DRAW is implied. When MODE=1, a relative DRAW is implied. When MODE=0, the coordinate (X, Y) specifies the absolute coordinate of the end point of a line to be drawn. When MODE=1, the coordinate specifies the offsets to be used in drawing a line from the current beam position to a new position.

DETAILED DESCRIPTION:

This routine converts the coordinate values specified to absolute screen coordinates and generates either an absolute or relative 2D draw graphic order. This graphic order is placed at the mark position of the currently opened page. Note that relative draws are restricted to 1/2 of the screen size.

8.3.4 T2D2D - Transform 2D to 2D

NAME: T2D2D

FUNCTION: Transform a page of graphic orders into a page which consists entirely of 2D graphic orders.

CALLING FORMAT: CALL T2D2D (IGRAPH, IPAG2D)

DESCRIPTION OF PARAMETERS:

IGRAPH = An integer variable supplied by the user which specifies the number of the graphic page which is to be transformed. IGRAPH must be in the range:

1 <IGRAPH <256, IGRAPH≠IPAG2D

IPAG2D = An integer variable supplied by the user which specifies the number of the 2D page in which the transformed graphic orders are to be placed. IPAG2D must be in the range:

1 <IPAG2D <256, IPAG2D≠IGRAPH

DETAILED DESCRIPTION:

This routine sets up the coordinate converter to perform a graphic page to 2D transformation on the page of graphic instructions specified by IGRAPH. The Coordinate Converter is started and a block of transformed graphic instructions is output to the page specified by IPAG2D. Note that unpredictable results will occur if the 2D output file is not large enough to accommodate the transformed graphic instructions. Refer to Appendix D of the Graphic 7 Fortran Support Package (FSP) User's Manual to determine the necessary output file size.

8.3.5 MTRX2D - Compute and Replace Matrix Parameters

NAME: MTRX2D

FUNCTION: Compute matrix parameters and generate coordinate converter instruction to update the matrix parameters.

CALLING FORMAT: CALL MTRX2D (ARRAY)

DESCRIPTION OF PARAMETERS:

ARRAY = A 7 element real array supplied by the user which specifies the scaling, translation, and rotation factors necessary for computing the matrix parameters. The array parameters and their valid ranges are specified below:

ARRAY Element	Definition	Range
1	X-Pre Translation	$-(XU-XL) \leq XPRE \leq +(XU-XL)$
2	Y-Pre Translation	$-(YU-YL) \leq YPRE \leq +(YU-YL)$
3	X Scale Factor	$1/256 \leq XSC < 128$
4	Y Scale Factor	$1/256 \leq YSC < 128$
5	Z Rotation (X-Y Plane)	any real number (in radians)
6	X-Post Translation	$-(XU-XL) \leq XPOST \leq +(XU-XL)$
7	Y-Post Translation	$-(YU-YL) \leq YPOST \leq +(YU-YL)$

The variables used above are defined in the SCALE subroutine description.

e.g. CALL SCALE (XL, YL, XU, YU)

DETAILED DESCRIPTION:

This routine uses the array parameters passed to compute new matrix parameters. These parameters are computed as indicated in Appendix D and are entered into the CC2DBL of the currently opened page, replacing the previous matrix parameters. This routine must be called each time the user wishes to change translation, scaling, or rotation factors.

8.3.6 VI2DBX - Update Viewbox

NAME: VI2DBX

FUNCTION: Update viewbox boundaries in the currently opened page.

CALLING FORMAT: CALL VI2DBX (LV, RV, BV, TV)

DESCRIPTION OF PARAMETERS:

LV = Real variable supplied by the user which specifies the viewbox left boundary (minimum X). The valid range for LV is: XL<LV<RV

RV = Real variable supplied by the user which specifies the viewbox right boundary (maximum X). The valid range for RV is: LV<RV<XU

BV = Real variable supplied by the user which specifies the viewbox bottom boundary (minimum Y). The valid range for BV is: YL<BV<TV

TV = Real variable supplied by the user which specifies the viewbox top boundary (maximum Y). The valid range for TV is: BV<TV<YU

The parameters LV, RV, BV and TV are all specified in the user coordinate system.

DETAILED DESCRIPTION:

This routine updates the viewbox boundaries in the currently opened page. The CC2DBL viewbox parameters, generated by a previous call to the CC2DBL routine, are updated. This routine must be called in order to change the viewbox.

APPENDIX 'A'

CCBLK MATRIX ELEMENT DEFINITIONS

NEW CCBLK MATRIX (4 * 3)

Element

$$1,1 = \frac{Sx \cos Ay \cos Az}{64}$$

$$1,2 = \frac{Sx \cos Ay \sin Az}{64}$$

$$1,3 = \frac{-Sx \sin Ay}{64}$$

$$2,1 = \frac{+Sy (\sin Ax \sin Ay \cos Az - \cos Ax \sin Az)}{64}$$

$$2,2 = \frac{+Sy (\sin Ax \sin Ay \sin Az + \cos Ax \cos Az)}{64}$$

$$2,3 = \frac{Sy \sin Ax \cos Ay}{64}$$

$$3,1 = \frac{Sz (\cos Ax \sin Ay \cos Az + \sin Ax \sin Az)}{64}$$

$$3,2 = \frac{Sz (\cos Ax \sin Ay \sin Az - \sin Ax \cos Az)}{64}$$

$$3,3 = \frac{Sz \cos Ax \cos Ay}{64}$$

$$4,1 = [Tx\ Sx \cos Ay + (Ty\ Sy \sin Ax + Tz\ Sz \cos Ax) \sin Ay] \cos Az - (Ty\ Sy \cos Ax - Tz\ Sz \sin Ax) \sin Az + Txp$$

$$4,2 = [Tx\ Sx \cos Ay + (Ty\ Sy \sin Ax + Tz\ Sz \cos Ax) \sin Ay] \sin Az + (Ty\ Sy \cos Ax - Tz\ Sz \sin Ax) \cos Az + Typ$$

$$4,3 = -Tx\ Sx \sin Ay + (Ty\ Sy \sin Ax + Tz\ Sz \cos Ax) \cos Ay + Tzp$$

APPENDIX 'B'

CCBLK FORMAT

WORD #	COMMAND	FORMAT	DESCRIPTION
0	LBOX	LBOX	Load viewbox parameters
1		LV	Left Boundary
2		BV	Bottom Boundary
3		NV	Near Boundary
4		RV	Right Boundary
5		TV	Top Boundary
6		FV	Far Boundary
7		Xa	X - Eye Point
8		Ya	Y - Eye Point
9		Za	Z - Eye Point
10	LMTX	LMTX	Load matrix parameters
11		M11	Matrix Elements
12		M12	
13		M13	
14		M21	
15		M22	
16		M23	
17		M31	
18		M32	

APPENDIX 'B'

CCBLK FORMAT (Cont)

WORD #	COMMAND	FORMAT	DESCRIPTION
19		M33	
20		M41	
21		M42	
22		M43	

APPENDIX 'C'

ADVANCED 3D APPLICATIONS

The routines described provide basic 3D graphic capabilities and the operations performed are not cumulative. The advanced application may require that matrix concatenations be performed in the host computer. In this case, a concatenated matrix may be sent to the Graphic-7 by using an FSP REFDAT command. In this way the user can directly update the CCBLK matrix parameters. The CCBLK viewbox and view point parameters can be updated similarly.

APPENDIX 'D'

CC2DBL MATRIX ELEMENT DEFINITIONS

NEW CC2DBL MATRIX (3 x 2)

ELEMENT

$$1,1 = \frac{Sx\ CosAxy}{64}$$

$$1,2 = \frac{Sx\ SinAxy}{64}$$

$$2,1 = \frac{Sy\ SinAxy}{64}$$

$$2,2 = \frac{Sy\ CosAxy}{64}$$

$$3,1 = \frac{-Xc\ Sx\ CosAxy - Yc\ Sy\ SinAxy + Xc + Tx}{64}$$

$$3,2 = \frac{-Xc\ Sx\ SinAxy - Yc\ Sy\ CosAxy + Yc + Ty}{64}$$

APPENDIX 'E'

CC2DBL FORMAT

WORD #	COMMAND	FORMAT	DESCRIPTION
0	LBOX	LBOX	Load viewbox parameters
1		LV	Left Boundary
2		BV	Bottom Boundary
3		RV	Right Boundary
4		TV	Top Boundary
5	LMTX	LMTX	Load matrix parameters
6		M11	Matrix Elements
7		M12	
8		M21	
9		M22	
10		M31	
11		M32	

Composite Matrix

Translation

$$X1 = X\phi + Tx$$

$$Y1 = Y\phi + Ty$$

$$\begin{bmatrix} X1 & Y1 & W1 \end{bmatrix} = \begin{bmatrix} Xo & Yo & Wo \end{bmatrix} \begin{bmatrix} 1 & \emptyset & 0 \\ \emptyset & 1 & 0 \\ Tx & Ty & 1 \end{bmatrix}$$

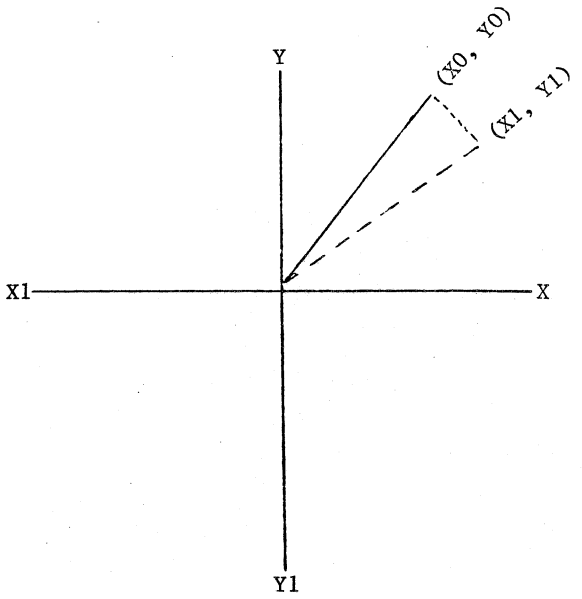
Scaling

$$X1 = Sx\ Xo$$

$$Y1 = Sy\ Yo$$

$$\begin{bmatrix} X1 & Y1 & W1 \end{bmatrix} = \begin{bmatrix} Xo & Yo & Wo \end{bmatrix} \begin{bmatrix} Sx & \emptyset & \emptyset \\ \emptyset & Sy & \emptyset \\ \emptyset & \emptyset & 1 \end{bmatrix}$$

ROTATION



$$X1 = Xo\ Cosx + Yo\ Sinx$$

$$Y1 = Xo\ Sinx + Yo\ Cosx$$

For rotation in Y-Y plane (clockwise)

Axy = angle of rotation in X-Y plane

$$X1 = Xo\ CosAxy + Yo\ SinAxy$$

$$Y1 = Xo\ SinAxy + Yo\ CosAxy$$

$$\begin{bmatrix} X1 & Y1 & W1 \end{bmatrix} = \begin{bmatrix} Xo & Yo & Wo \end{bmatrix} \begin{bmatrix} CosAxy & SinAxy & \emptyset \\ -SinAxy & CosAxy & \emptyset \\ \emptyset & \emptyset & 1 \end{bmatrix}$$

Method III

Order of Matrix Multiplication

1. Pre-translate to origin
2. Scale
3. Rotate
4. Post-translate back to original coordinates
5. Translate to new coordinates

Pre-Trans Scale

$$\begin{bmatrix} 1 & \emptyset & \emptyset \\ \emptyset & 1 & \emptyset \\ -Xc & -Yc & 1 \end{bmatrix} \begin{bmatrix} Sx & \emptyset & \emptyset \\ \emptyset & Sy & \emptyset \\ \emptyset & \emptyset & 1 \end{bmatrix} = \begin{bmatrix} Sx & \emptyset & \emptyset \\ \emptyset & Sy & \emptyset \\ -XcSx & -YcSy & 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} Sx & \emptyset & \emptyset \\ \emptyset & Sy & \emptyset \\ -XcSx & -YcSy & 1 \end{bmatrix} \begin{bmatrix} CosAxy & SinAxy & \emptyset \\ -SinAxy & CosAxy & \emptyset \\ \emptyset & \emptyset & 1 \end{bmatrix} = \begin{bmatrix} SxCosAxy & SxSinAxy & \emptyset \\ -SySinAxy & SyCosAxy & \emptyset \\ -XcSxCosAxy & -XcSxSinAxy & 1 \\ +YcSySinAxy & -YcSyCosAxy & \end{bmatrix}$$

Post-Trans

$$\begin{bmatrix} SxCosAxy & SxSinAxy & \emptyset \\ -SySinAxy & SyCosAxy & \emptyset \\ -XcSxCosAxy & -XcSxSinAxy & 1 \\ +YcSySinAxy & -YcSyCosAxy & \end{bmatrix} \begin{bmatrix} 1 & \emptyset & \emptyset \\ \emptyset & 1 & \emptyset \\ Xc & Yc & 1 \end{bmatrix} = \begin{bmatrix} SxCosAxy & SxSinAxy & \emptyset \\ -SySinAxy & SyCosAxy & \emptyset \\ -XcSxCosAxy & -XcSxSinAxy & 1 \\ +YcSySinAxy & -YcSyCosAxy & \\ +Xc & +Yc & \end{bmatrix}$$

$$\begin{bmatrix} SxCosAxy & SxSinAxy & \emptyset \\ -SySinAxy & SyCosAxy & \emptyset \\ -XcSxCosAxy & -XcSxSinAxy & 1 \\ +YcSySinAxy & -YcSyCisAxy & \\ +Xc & +Yc & \end{bmatrix} \begin{bmatrix} 1 & \emptyset & \emptyset \\ \emptyset & 1 & \emptyset \\ Tx & Ty & 1 \end{bmatrix} = \begin{bmatrix} SxCosAxy & SxSinAxy & \emptyset \\ -SySinAxy & SyCosAxy & \emptyset \\ -XcSxCosAxy & -XcSxSinAxy & 1 \\ +YcSySinAxy & -YcSyCosAxy & \\ +Xc +Tx & +Yc +Ty & \end{bmatrix}$$

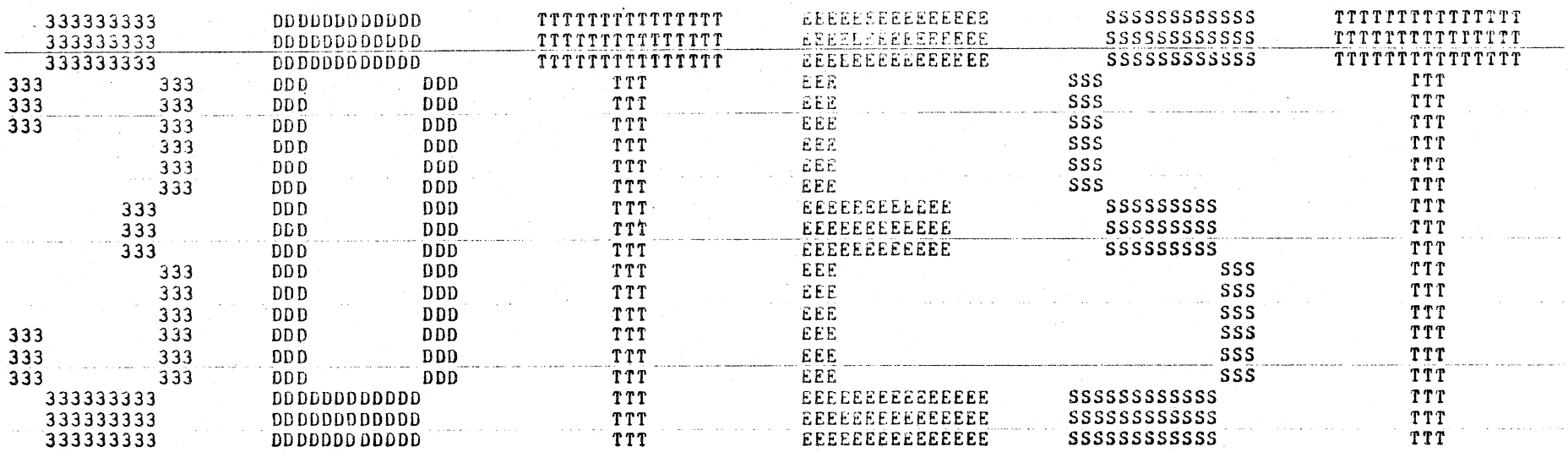
Change 1

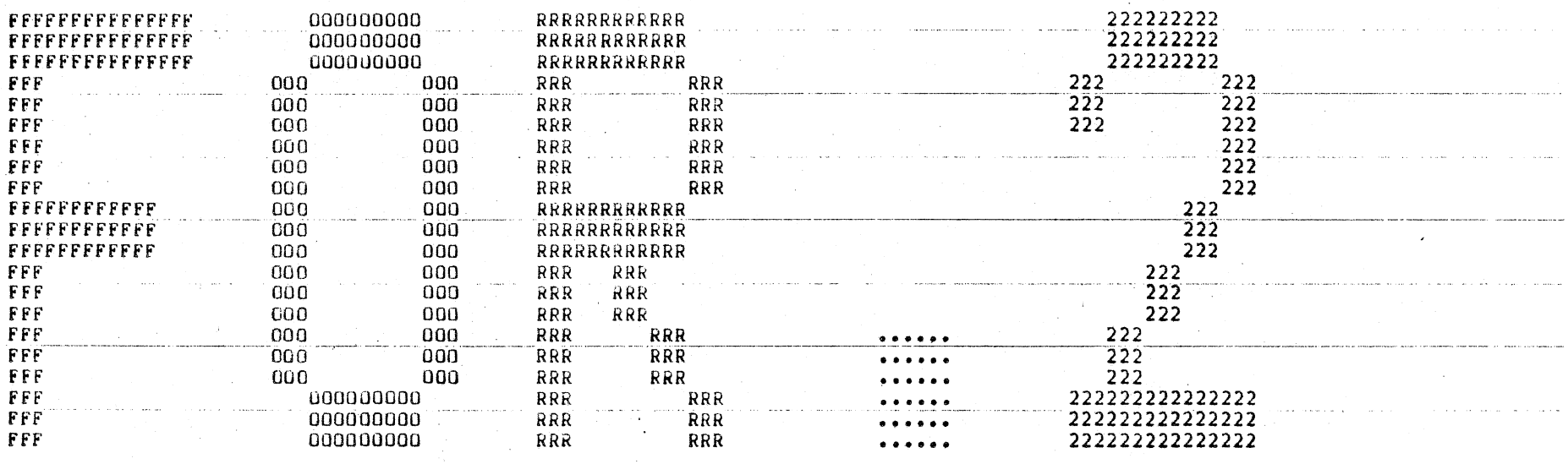
E3/(E-4 blank)

APPENDIX 'F'

FSP SAMPLE PROGRAMS - 2D/3D

- 3DTEST.FOR - 3D SAMPLE
- TEST2D.FOR - 2D SAMPLE





START Job FSP01 Req #2446 for E00018 Date 12-May-81 11:45:42 Monitor: SANDERS ASSOCIATES INC., TOPS-20 Monitor *START*
File DSKB:<E00018>3DTEST.FOR.2, created: 29-Apr-80 11:38:33, printed: 12-May-81 11:48:20
Job parameters: Request created:12-May-81 11:19:46 Page limit:216 Forms:NORMAL Account:60777
File parameters: Copy: 1 of 1 Spacing:SINGLE File format:ASCII Print mode:ASCII

```
C  MARK.FTN
C
      DIMENSION IARRAY(3),ARRAY(12)
      DATA IARRAY/1000,1000,1000/
      CALL GSS4(5,0,2)
      CALL LAYOUT(3,IARRAY)
      CALL INIT3D
      CALL SCALE(0.,0.,200.,200.)
      CALL SCAL3D(0.,100.)
      CALL ADDREF(2)
      CALL CCBLK
C
C  DRAW A CUBE 50 UNITS ON A SIDE WITH CENTER AT (100,100,45)
C
      CALL MOVE3D(75.,75.,20.,0)
      CALL DRAW3D(50.,0.,0.,1)
      CALL DRAW3D(0.,50.,0.,1)
      CALL DRAW3D(-50.,0.,0.,1)
      CALL DRAW3D(0.,-50.,0.,1)
      CALL MOVE3D(0.,0.,50.,1)
      CALL DRAW3D(50.,0.,0.,1)
      CALL DRAW3D(0.,50.,0.,1)
      CALL DRAW3D(-50.,0.,0.,1)
      CALL DRAW3D(0.,-50.,0.,1)
      CALL DRAW3D(0.,0.,-50.,1)
      CALL MOVE3D(50.,0.,50.,1)
      CALL DRAW3D(0.,0.,-50.,1)
      CALL MOVE3D(0.,50.,50.,1)
      CALL DRAW3D(0.,0.,-50.,1)
      CALL MOVE3D(-50.,0.,50.,1)
      CALL DRAW3D(0.,0.,-50.,1)
      CALL MOVE3D(100.,100.,45.,0)
      CALL SETEXT('MARK',4)
      ANGLE=0.
      RAD=180./3.14159265
      CALL ADDREF(3)
      CALL ERASEP
      CALL ADDREF(1)
      CALL PICTUR(3)
      CALL ADDREF(2)
      CALL VIEWPT(100.,100.,-100.)
      CALL T3D2D(2,3)
5     DO 2 I=1,12
2     ARRAY(I)=0.
      ARRAY(4)=1
      ARRAY(5)=1
      ARRAY(6)=1
6     CALL EVENT(I)
      IF(I .NE. 4) GO TO 6
      CALL GETKEY(I,IKEY)
      IKEY=IKEY-16
      IF(IKEY .EQ. 0) GO TO 99
      GO TO (100,200,300,400,500,600),IKEY
      GO TO 6
C
C  THIS SECTION CHANGES THE VIEWPOINT FROM (100,0,100) TO (100,200,100)
C  IN CONTINUOUS STEPS
```

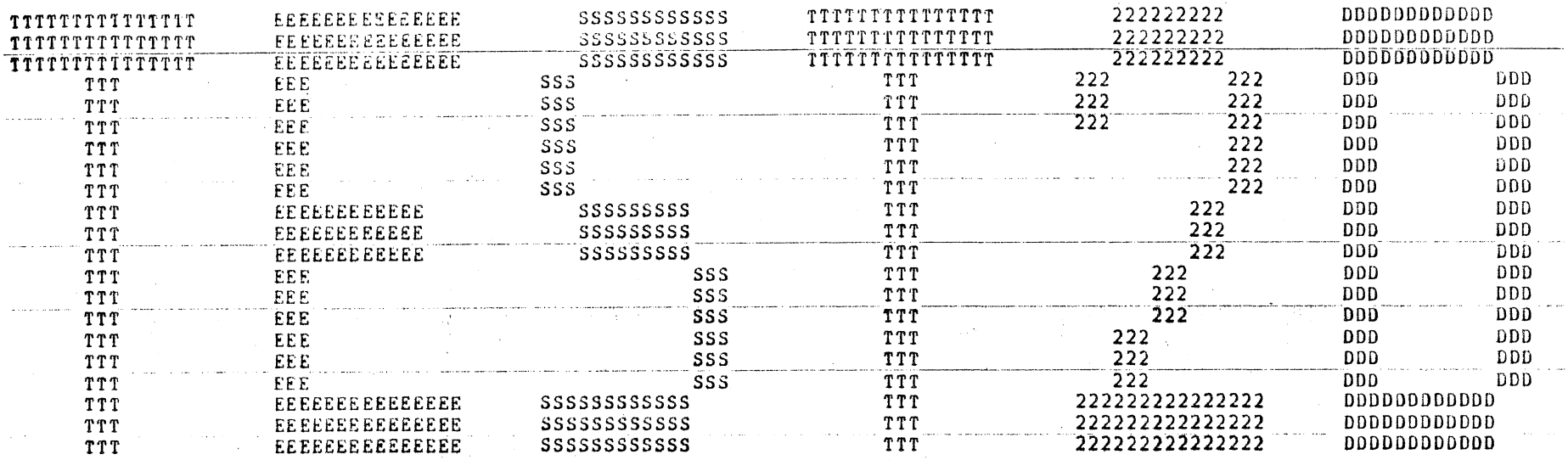
```
        Y=100.+J
        CALL VIEWPT(100.,Y,-100.)
110     CALL T3D2D(2,3)
        CALL VIEWPT(100.,100.,-100.)
        GO TO 5
C
C  CHANGES THE SCALING FROM 1 TO 1/64, 1/64 TO 64, AND FROM 64 TO 1
C
200     DO 210 J=1,64
        X=1.0/J
        ARRAY(4)=X
        ARRAY(5)=X
        ARRAY(6)=X
        CALL MTRX3D(ARRAY)
210     CALL T3D2D(2,3)
        DO 220 J=64,1,-1
        X=1.0/J
        ARRAY(4)=X
        ARRAY(5)=X
        ARRAY(6)=X
        CALL MTRX3D(ARRAY)
220     CALL T3D2D(2,3)
        DO 230 J=1,64
        X=J
        ARRAY(4)=X
        ARRAY(5)=X
        ARRAY(6)=X
        CALL MTRX3D(ARRAY)
230     CALL T3D2D(2,3)
        DO 240 J=64,1,-1
        X=J
        ARRAY(4)=X
        ARRAY(5)=X
        ARRAY(6)=X
        CALL MTRX3D(ARRAY)
240     CALL T3D2D(2,3)
        GO TO 5
C
C  TRANSLATES THE OBJECT FIRST LEFT AND RIGHT IN THE X DIRECTION,
C  THEN UP AND DOWN IN THE Y DIRECTION, AND LASTLY BACK AND FORTH
C  IN THE Z DIRECTION
C
300     DO 330 J=1,3
        ARRAY(1)=0.
        ARRAY(2)=0.
        ARRAY(3)=0.
        DO 310 I=0,-100,-2
        ARRAY(J)=I
        CALL MTRX3D(ARRAY)
310     CALL T3D2D(2,3)
        DO 320 I=-100,100,2
        ARRAY(J)=I
        CALL MTRX3D(ARRAY)
320     CALL T3D2D(2,3)
        DO 330 I=100,0,-2
        ARRAY(J)=I
        CALL MTRX3D(ARRAY)
330     CALL T3D2D(2,3)
```

```
C  ROTATE THE OBJECT AROUND ITS CENTER
C
400      ARRAY(1)=0.
         ARRAY(2)=0.
         ARRAY(3)=-45.
         ARRAY(10)=0.
         ARRAY(11)=0.
         ARRAY(12)=45.
         DO 410 J=0,360,2
         ARRAY(7)=J/RAD
         CALL MTRX3D(ARRAY)
410      CALL T3D2D(2,3)
         DO 420 J=0,360,2
         ARRAY(8)=J/RAD
         CALL MTRX3D(ARRAY)
420      CALL T3D2D(2,3)
         DO 430 J=0,360,2
         ARRAY(9)=J/RAD
         CALL MTRX3D(ARRAY)
430      CALL T3D2D(2,3)
         GO TO 5
C
C  SCALE THE CUBE BY A FACTOR OF 1/4, ROTATE IT AROUND THE X-AXIS
C  RELATIVE TO ITS CENTER WHILE MOVING THE ENTIRE CUBE IN A
C  CIRCULAR MOTION AROUND THE CENTER OF THE SCREEN
C
500      ARRAY(1)=0.
         ARRAY(2)=0.
         ARRAY(3)=-45.
         ARRAY(4)=.25
         ARRAY(5)=.25
         ARRAY(6)=.25
         ARRAY(10)=0.
         ARRAY(11)=0.
         ARRAY(12)=45.
         DO 510 J=0,360
         THETA=J/RAD
         THETA2=(J*4.)/RAD
         ARRAY(7)=THETA2
         ARRAY(10)=100.*COS(THETA)
         ARRAY(11)=100.*SIN(THETA)
         CALL MTRX3D(ARRAY)
510      CALL T3D2D(2,3)
         GO TO 5
C
C  CHANGE THE VIEWBOX FIRST BY COMPRESSING THE X CLIPPING BOUNDARY,
C  THEN THE Y, AND LASTLY THE Z

C
600      VL=0.
         VR=200.
         VB=0.
         VT=200.
         VN=0.
         VF=400.
         ARRAY(4)=4.
         ARRAY(5)=4.
         ARRAY(6)=4.
```

Change F5

```
        DO 610 I=0,98,2
        VL=I
        VR=200.-I
        CALL VIEWBX(VL,VR,VB,VT,VN,VF)
610     CALL T3D2D(2,3)
        DO 620 I=98,0,-2
        VL=I
        VR=200.-I
        CALL VIEWBX(VL,VR,VB,VT,VN,VF)
620     CALL T3D2D(2,3)
        DO 630 I=0,98,2
        VB=I
        VT=200.-I
        CALL VIEWBX(VL,VR,VB,VT,VN,VF)
630     CALL T3D2D(2,3)
        DO 640 I=98,0,-2
        VB=I
        VT=200.-I
        CALL VIEWBX(VL,VR,VB,VT,VN,VF)
640     CALL T3D2D(2,3)
        DO 650 I=0,49
        VN=I*4.
        VF=400.-(I*4.)
        CALL VIEWBX(VL,VR,VB,VT,VN,VF)
650     CALL T3D2D(2,3)
        DO 660 I=49,0,-1
        VN=I*4.
        VF=400.-(I*4.)
        CALL VIEWBX(VL,VR,VB,VT,VN,VF)
660     CALL T3D2D(2,3)
        VF=(32767./511.)*100.
        CALL VIEWBX(VL,VR,VB,VT,VN,VF)
        GO TO 5
C
C  END THE PROGRAM
C
99      CALL THEEND
        STOP
        END
```



START Job FSP01 Req #2446 for E00018 Date 12-May-81 11:45:42 Monitor: SANDERS ASSOCIATES INC., TOPS-20 Monitor *START*
File DSKB:<E00018>TEST2D.FOR.1, created: 20-Feb-81 10:30:39, printed: 12-May-81 11:48:06
Job parameters: Request created:12-May-81 11:19:46 Page limit:216 Forms:NORMAL Account:60777
File parameters: Copy: 1 of 1 Spacing:SINGLE File format:ASCII Print mode:ASCII

```
C  MARK.FTN
C
      DIMENSION IARRAY(3),ARRAY(12)
      DIMENSION IOUTB(50),INB(70),IEVENT(150),LAMPA(4,2)
      DIMENSION IMARK(256),LPAGES(256),IPEVNT(2),IBNK(8),IPBNK(4)
      DIMENSION IUPED(4),IPEDA(4),IUSL(4),LDZS(4),LDZSA(4)
      COMMON /TERMB/ IOUTB,INB,IFAC,IUNIT,IEVENT,LAMPA,IPEVNT,INIFAC
      COMMON /PVMD/ ILXP,IHXP,ILYP,IHYP,IBFPTR,IRL,LBFPTR,LRL,IPFLAG
      COMMON /COORD/ XLOW,XHI,YLOW,YHI,IXB,IYB,IADR,IOPT,INDH
      COMMON /LAYOT/ LPAGES,IMARK,IPAGEC,LOCK,MAXPAG,MODE
      COMMON /MAST/ IERRA,LDZS,LDZSA
      COMMON /PERIPH/ IPBNK,IUPED,IPEDA,IUSL
      COMMON /LMEM/ IBNK,ICBANK,ISBANK,ISWORD
      DATA IARRAY/1000,1000,1000/
      CALL GSS4(5,0,2)
      CALL LAYOUT(3,IARRAY)
      CALL SCALE(0.,0.,200.,200.)
      CALL ADDREF(2)
      CALL CC2DBL
C
C  DRAW A CUBE 50 UNITS ON A SIDE WITH CENTER AT (100,100,45)
C
      CALL MOVE2D(75.,75.,0)
      CALL DRAW2D(50.,0.,1)
      CALL DRAW2D(0.,50.,1)
      CALL DRAW2D(-50.,0.,1)
      CALL DRAW2D(0.,-50.,1)
      CALL MOVE2D(90.,100.,0)
C     CALL SETEXT('2D TE',5)
      ANGLE=0.
      RAD=180./3.14159265
      CALL ADDREF(3)
      CALL ERASEP
      CALL ADDREF(1)
      CALL PICTUR(3)
      CALL ADDREF(2)
      CALL MOVE2D(100.,100.,0)
      CALL T2D2D(2,3)
      IPNT=0
5     DO 2 I=1,7
2     ARRAY(I)=0.
      ARRAY(3)=1
      ARRAY(4)=1
6     CALL EVENT(I)
      IF(I .NE. 4) GO TO 6
      CALL GETKEY(I,IKEY)
      IKEY=IKEY-16
      IF(IKEY .EQ. 0) GO TO 99
      GO TO (99,200,300,400,500,600,700,800),IKEY
800   IPNT=IPNT-1
      GO TO 701
700   IPNT=IPNT+1
701   ARRAY(3)=IPNT
      ARRAY(4)=IPNT
      CALL MTRX2D(ARRAY)
      CALL T2D2D(2,3)
      GO TO 5
      GO TO 6
```

```
C  CHANGES THE SCALING FROM 1 TO 1/50, 1/50 TO 50, AND FROM 50 TO 1
C
200     DO 210 J=1,50
        X=1.0/J
        ARRAY(3)=X
        ARRAY(4)=X
        CALL MTRX2D(ARRAY)
210     CALL T2D2D(2,3)
        DO 220 J=50,1,-1
        X=1.0/J
        ARRAY(3)=X
        ARRAY(4)=X
        CALL MTRX2D(ARRAY)
220     CALL T2D2D(2,3)
        DO 230 J=1,50
        X=J
        ARRAY(3)=X
        ARRAY(4)=X
        CALL MTRX2D(ARRAY)
230     CALL T2D2D(2,3)
        DO 240 J=50,1,-1
        X=J
        ARRAY(3)=X
        ARRAY(4)=X
        CALL MTRX2D(ARRAY)
240     CALL T2D2D(2,3)
        GO TO 5
C
C  TRANSLATES THE OBJECT FIRST LEFT AND RIGHT IN THE X DIRECTION,
C  THEN UP AND DOWN IN THE Y DIRECTION, AND LASTLY BACK AND FORTH
C  IN THE Z DIRECTION

C
300     DO 330 J=1,2
        ARRAY(1)=0.
        ARRAY(2)=0.
        DO 310 I=0,-80,-2
        ARRAY(J)=I
        CALL MTRX2D(ARRAY)
310     CALL T2D2D(2,3)
        DO 320 I=-80,80,2
        ARRAY(J)=I
        CALL MTRX2D(ARRAY)
320     CALL T2D2D(2,3)
        DO 330 I=80,0,-2
        ARRAY(J)=I
        CALL MTRX2D(ARRAY)
330     CALL T2D2D(2,3)
        GO TO 5
C
C  ROTATE THE OBJECT AROUND ITS CENTER
C
400     ARRAY(1)=0.
        ARRAY(2)=0.
        ARRAY(6)=0.
        ARRAY(7)=0.
        DO 410 J=0,360,2
        ARRAY(5)=J/RAD
```

```
410     CALL T2D2D(2,3)
        GO TO 5
C
C  SCALE THE CUBE BY A FACTOR OF 1/4, ROTATE IT AROUND THE X-AXIS
C  RELATIVE TO ITS CENTER WHILE MOVING THE ENTIRE CUBE IN A
C  CIRCULAR MOTION AROUND THE CENTER OF THE SCREEN
C
500     ARRAY(1)=0.
        ARRAY(2)=0.
        ARRAY(3)=.25
        ARRAY(4)=.25
        ARRAY(6)=0.
        ARRAY(7)=0.
        DO 510 J=0,360
        THETA=J/RAD
        THETA2=(J*2.)/RAD
        ARRAY(5)=THETA2
        ARRAY(6)=100.*COS(THETA)
        ARRAY(7)=100.*SIN(THETA)
        CALL MTRX2D(ARRAY)
510     CALL T2D2D(2,3)
        GO TO 5
C
C  CHANGE THE VIEWBOX FIRST BY COMPRESSING THE X CLIPPING BOUNDARY,
C  THEN THE Y, AND LASTLY THE Z

C
600     VL=0.
        VR=150.
        VB=0.
        VT=150.
        ARRAY(3)=4.
        ARRAY(4)=4.
        CALL MTRX2D(ARRAY)
        DO 610 I=0,98,2
        VL=I
        VR=150.-I
        CALL VI2DBX(VL,VR,VB,VT)
610     CALL T2D2D(2,3)
        DO 620 I=98,0,-2
        VL=I
        VR=150.-I
        CALL VI2DBX(VL,VR,VB,VT)
620     CALL T2D2D(2,3)
        DO 630 I=0,98,2
        VB=I
        VT=150.-I
        CALL VI2DBX(VL,VR,VB,VT)
630     CALL T2D2D(2,3)
        DO 640 I=98,0,-2
        VB=I
        VT=150.-I
        CALL VI2DBX(VL,VR,VB,VT)
640     CALL T2D2D(2,3)
        GO TO 5
C
C  END THE PROGRAM
C
99      CALL THEEND
```

Name:_____

Company:_____

Address:_____

_____

Telephone: [ ]_____

Date:_____

Sanders Equipment_____

Part Number_____

Software/Firmware System_____

Version _____

Host computer_____

Host operating system_____ Version_____

Host-GRAPHIC 7 interface_____

My problem is: hardware ☐ software ☐

firmware ☐ manual ☐

Description of problem (or suggestion for improvement):

Related tech manual number _____

THE INTENT AND PURPOSE OF THIS PUBLICATION IS TO PROVIDE ACCURATE AND MEANINGFUL INFORMATION TO SUPPORT EQUIPMENT MANUFACTURED BY SANDERS ASSOCIATES, INC. YOUR COMMENTS AND SUGGESTIONS ARE REQUESTED.

PLEASE USE THE FORM ON THE REVERSE SIDE TO REPORT ANY PROBLEMS YOU HAVE HAD WITH THIS PUBLICATION OR THE EQUIPMENT IT DESCRIBES.

FOLD FOLD

FIRST CLASS
PERMIT NO. 568
NASHUA, N.H.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by

Sanders Associates, Inc.
Information Products Division
Daniel Webster Highway South
Nashua, New Hampshire 03061

ATTN: DEPARTMENT 1-2894 (NHQ 1-447)

FOLD FOLD

SANDERS