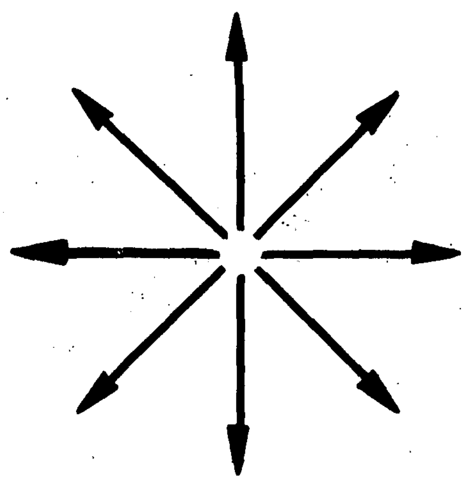
THE CHALLENGER CHARACTER GRAPHICS REFERENCE MANUAL



Aug. 1978

TABLE OF CONTENTS

SECTION 1 INTRODUCTION

1.1 DEFINITION OF CHARACTER GRAPHICS

The term "graphics" broadly describes various types of pictures, line drawings, or other related visual media. In the context of computing, graphics is the process of generating these visual effects on some display device via software control.

Computer graphics are generally displayed on a video screen This allows rapid manipulation of the display for real-time screen updating, animations, etc. Printers are not practical for any but the least involved graphics displays because they lack the high speed capabilities of the video display.

"Character" graphics is the creation of these visuals through the arrangement or movement of alphabetic, numeric, and special-purpose characters on the video screen. Examples of these special characters are various geometric shapes, vehicle outlines (planes, cars, etc.), and gaming symbols such as playing card suits. Non-character graphics rely on illuminating individual dots or squares on the screen, andhave no specialized character capabilities.

1.2 APPLICATIONS

The applications of computerized graphics fall into three general categories: mathematics, gaming, and free-standing graphics. These are briefly outlined as follows:

1. Mathematics - graphs, charts, histograms, plotting, graphic data representation.
2. Gaming - arcade, video score keeping, playing fields, game boards, simulations.
3. Free-standing graphics - animations, complex non-moving images, and kaleidoscopic displays.

1.3 BRIEF DESCRIPTION - OSI GRAPHICS CONCEPTS

Ohio Scientific microcomputer systems employing either the 600 board (complete single-board computer), or the 540 Video board and 542 Keyboard combination offer advanced graphics capabilities. The 540 and 600 boards utilize the CG-4 character generator ROM which stores data for 256 gaphics characters. These include upper and lower case alphabets, numerals, punctuation, and over 160 gaming elements. The 542 and 600 boards contain special software-scanned (polled) keyboards which make full use of the graphics characters and alphabet provided by the CG-4 ROM.

In graphics applications, a particular character is called to the screen by POKEing the character's code number to the address of the video memory location where it is to be displayed. The code numbers are structured around the standard ASCII system, but incorporate extra codes to accomodate the additional non-standard characters. During normal entry of alphanumeric data, all alphabetics and numerals are available to the user through conventional keystrokes. Table 1-1 at the end of this manual shows outlines and codes for the 256 characters provided by the CG-4 firmware.

Table 1-2 provides data on the Ohio Scientific CHALLENGER series microcomputers which utilize the model 540, 542, and 600 boards. Conversion data is included for modifying older Ohio Scientific microcomputers to perform advanced graphics or keyboard functions.

1.4 FEATURES - 540 AND 600 VIDEO

The Model 540 Video board contains 2K of memory dedicated to the video display. This gives a 32 line/64 column format. A guard band has been provided on the lateral edges of the screen so that a full 64 columns are visible. A minimum of 30 lines is normally visible on a true TV monitor. Where a modified television is used as a monitor, the vertical height control may require adjustment to display the maximum number of lines. Figure 1-1 at the end of this manual shows the memory mapping for the 540 Video board. Each square in the grid represents a location on the video screen.

TABLE 1-2 OSI MICROCOMPUTERS - GRAPHICS AND EXPANSION CAPABILITIES

Model	CPU Bd.	Key Bd.	Video Bd.	Format	Remarks
C1-P Super-board II		600		25 x 25	BASIC-in-ROM single board computer. Expandable to disk, 32K RAM with Model 610 interface and modifications.* Available factory configured for disk and 32K RAM.
CII-4P CII-8P	500 Rev D, or 502	542	540 with CG-4 ROM**	32 x 32+ 32 x 64	BASIC-in-ROM multi-board computer. Expandable to disk operation with slight modifications.* Available factory configured for disk operation.
All other Model Challengers or Conversions	400 500 Rev A 500 Rev B 500 Rev C	ASCII	440 540 without CG-4 ROM**	—	Most older OSI Challenger microcomputers can be converted to character graphics and/or polled keyboard operation.*

Notes:

*Required modifications vary. Consult dealer or factory for further data or authorized service.

+Software selectable via POKE statements.

**CG-4 ROM - 256 Characters including upper and lower case alpha, numerals, punctuation, game elements.

The 540 Video board is automatically reset to the 32/64 format when the system is powered up. However, the 540 board can also operate in a 32 line/32 column format. The contents of the memory location 56832(dec) determines the format. When an odd number from 0 to 255 is stored in this location, the format will be 32/64. When the contents of this location is even, format is 32/32. This location can be addressed and altered via a POKE statement, either in the immediate mode or under software control.

In the 32/32 format, the character addressing system is the same as that used in the 32/64 format. The major difference is that only the first 32 characters of each line are visible. The memory mapping for the 540 Video board in the 32/32 format is provided in Figure 1-2 (end of manual). The guard band and 30-line feature of the 32/64 format have been retained.

The model 600 board contains a 1K single-format graphics system. The video memory begins at D000(hex), or 53248(dec). However, the guard band feature has not been incorporated, and the visible character field consists of 25 lines of 25 columns. The first visible character in the upper left of the screen is accessed via address 53379(dec). The video memory map for the 600 board is shown in Figure 1-3 (end of manual).

SECTION 2 GRAPHICS IMPLEMENTATION

2.1 BASIC POKE STATEMENT

The OSI character graphics system utilizes the BASIC language "POKE" statement to access the video memory. The code for a desired graphics character is POKEd to a specific memory location. A typical POKE statement assumes the form:

```
POKE (address,base 10),(character code,base 10)
EXAMPLE: POKE 54832,1
```

The outline and code for each of the 256 graphics characters provided by the CG-4 ROM is given in Table 1-1 (end of manual).

2.2 PLOTTING TECHNIQUES

Using POKE statements, video memory maps, and graphics character outline/code number tables, the user can generate programs for involved still images. Several short examples are now provided for programming motion on the screen. These sample programs use the "square" character which is coded "161". The theories outlined can be expanded for more elaborate programming.

2.2.1 HORIZONTAL PLOTTING - 540 AND 600 BOARDS

The following programs are intended for a system employing a 540 Video board, and can be run in the 32/32 or 32/64 format. For systems using the 600 board, the video addresses 53699 and 53723 should be substituted for 53830 and 53870 respectively. Refer to the video memory maps, Figures 1-1, 1-2, and 1-3 as needed.

```
10 REM---DRAW LINE LEFT TO RIGHT---
20 FOR X=53830 TO 53870
30 POKE X,161
40 FOR T=0 TO 50:NEXT T:REM---"T" LOOP=TIME DELAY---
50 NEXT X
```

For a line from right to left, modify the program as follows:

```
10 REM---DRAW LINE RIGHT TO LEFT---
20 FOR X=53870 TO 53830 STEP -1:
25 REM---"X" LOOP COUNTS BACKWARD FROM 53870---
40 FOR T=0 TO 20:NEXT T:REM--FASTER "T" LOOP--
```

To move a single square from right to left, add:

```
45 POKE X,32
46 REM---32 IS THE CODE FOR A BLANK SPACE---
47 REM--SQUARE ERASED BEFORE NEXT ONE APPEARS--
```

To move a single character left to right, change:

```
20 FOR X=53830 TO 53870
```

2.2.2 VERTICAL AND ANGULAR PLOTTING - 540 BOARD

Vertical motion is achieved by POKEing characters into a vertical column. With the 540 Video board, contiguous vertical screen locations may be accessed via video address increments of +64 or -64. Specifically, the FOR-NEXT loop containing the video addresses STEPs by +64 or -64. This point is illustrated by the following program (refer to the video memory maps as needed).

```
10 REM---PLOTTING DOWN---
20 FOR X=54288 TO 54800 STEP 64
30 POKE X,161
40 FOR T=0 TO 50:NEXT T:REM--TIMING LOOP--
50 POKE X,32:REM---ERASES SQUARE---
60 NEXT X
```

To plot upward, change the following statements:

```
10 REM---PLOTTING UPWARD---
20 FOR X=54288 TO 53248 STEP -64
```

Angular motion upward or downward can be plotted by using a FOR-NEXT loop increment other than +64 or -64. Motion at various vertical angles is plotted by using the increments shown in Figure 2-1. Remember that to increment the video address by a negative number, the FOR-NEXT loop must STEP from a numerically greater address back to a lesser address.

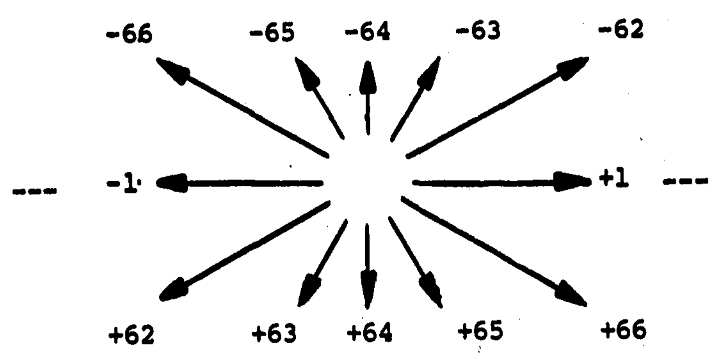


FIGURE 2-1. FOR-NEXT LOOP INCREMENTS - 540 VIDEO

2.2.3 VERTICAL AND ANGULAR PLOTTING - 600 BOARD

The same technique that is use for creating vertical movement with 540-based systems is used with the 600 board; that is, characters are POKEd into vertical rows on the screen. The difference is that the 600 video memory size is 1K rather than 2K as with the 540 board. Hence, the addresses which access the video screen locations are different. Contiguous vertical locations are plotted via FOR-NEXT loop increments of +32 or -32 (refer to the video memory map, Figure 1-3). This technique is demonstrated by the following program:

```
10 REM---PLOTTING DOWN---
20 FOR X=53741 TO 54171 STEP 32
30 POKE X,161
40 FOR T=0 TO 50:NEXT T:REM--TIMING LOOP---
50 POKE X,32:REM--ERASES SQUARE---
60 NEXT X
```

The program may be altered to make the square move upward by changing the following:

```
10 REM---PLOTTING UPWARD---
20 FOR X=54171 TO 53741 STEP -32
```

Angular motion upward or downward can be plotted by using a FOR-NEXT loop increment other than +32 or -32. Motion at various angles to the vertical is achieved by using the increments shown in Figure 2-2. When writing a FOR-NEXT loop for upward motion, remember that the loop must STEP from a numerically greater video address to a lesser address.

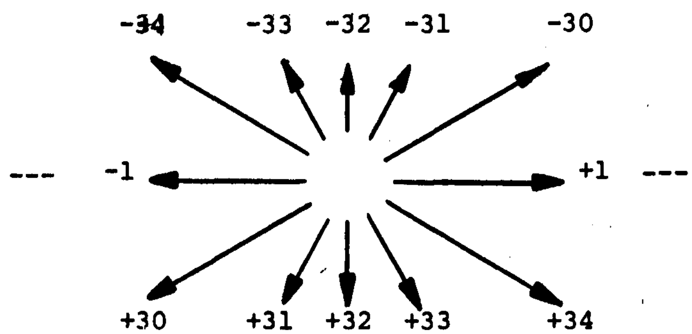


FIGURE 2-2. FOR-NEXT LOOP INCREMENTS - 600 VIDEO

2.3 SCREEN CLEAR TECHNIQUES

The video can be cleared by one of three methods. The first of these is a multiple scroll. The second is an extension of the BASIC POKE statement which fills the screen with blanks. The third is a fast machine language subroutine which gives instantaneous screen clears. Each method has unique features which suit various applications. Examples follow.

2.3.1 MULTIPLE SCROLL

This short FOR-NEXT loop will repetitively scroll the screen until it is blank. This method is fairly rapid but it is non-selective.

```
10 FOR X = 0 TO 32 : PRINT : NEXT X
```

2.3.2 BASIC POKE SCREEN CLEAR

The following program clears the screen by POKEing a blank space to video memory locations. Its advantage is that selective areas of the screen may be erased. It is somewhat slower than the multiple scroll.

```
10 REM--FIRST, FILL THE SCREEN WITH SQUARES--
20 FOR X=53248 TO 55295:REM--FOR 600 BOARD, USE--
21 REM--54272 INSTEAD OF 55295---
30 POKE X,161
40 NEXT X
50 REM--ERASE SCREEN NOW--
60 FOR X=53248 TO 55295:REM--FOR 600 BOARD, USE--
61 REM--54272 INSTEAD OF 55295---
70 POKE X,32
80 NEXT X
```

2.3.3 MACHINE LANGUAGE SCREEN CLEAR

The screen may be cleared very rapidly by utilizing a machine language subroutine. The BASIC program will execute the machine language subroutine when it encounters a statement containing the "USR" function, such as:

```
10 X=USR(X)
```

A requirement is that the highest 24 bytes of system RAM must be reserved for the subroutine. This is done after the system is powered up or reset by answering the prompt "MEMORY SIZE?" with a number equal to the address of the highest existing RAM minus 24 bytes. For example, on a system containing 4K RAM, 4095 - 24 = 4072 bytes. On a system containing 8K RAM, limit the memory size to 8191-24, or 8167 bytes. The BASIC language program then stores the decimal equivalents of the machine language op-codes in these 24 reserved locations. A general-purpose program for a 4K RAM system follows.

```
10 RESTORE : K=1024 : N=4 : B=K*N : C=B/256-1
15 POKE 11,232:POKE 12,C
20 FOR P=B-24 TO B-1:READ M:POKE P,M:NEXT P
25 POKE B-2,C:POKE B-10,C
30 DATA 169,32,160,8,162,0,157,0,208
40 DATA 232,208,250,238
50 DATA 240,15,136,208,244,169,208
60 DATA 141,240,15,96
```

NOTES: 1. "N" = number of K of system RAM. On an 8K system N=8; on a 16K system N=16, etc.
2. "P" = the range of addresses to be POKEd with DATA.
3. "M" = decimal values of op-codes stored as DATA.

To test the program, enter it and then the following additional statements:

```
100 REM--FIRST FILL THE SCREEN WITH SQUARES--
110 FOR X=53248 TO 55295:REM--FOR 600 BOARD,--
120 REM--USE 54272 INSTEAD OF 55295--
130 POKE X,161
140 NEXT X
150 FOR T=0 TO 250:NEXT T:REM--TIMING LOOP--
160 X=USR (X):REM--CALLS UP MACHINE LANGUAGE--
```

2.4 DISK SYSTEM GRAPHICS TECHNIQUES

The rapid accessibility of stored data and the overall flexibility of a disk operating system permit several unique graphics operations. Two of these are outlined here. The first technique is the direction of PRINT statement data to some screen location other than to the bottom of the screen. The second is the direct transfer of the entire video memory contents to the disk, or from disk to video RAM. The example routines provided are for compatible with the Ohio Scientific OS-65D disc operating systems, versions 2.0 and 3.0. When incorporating these routines into larger programs, alter the routine line numbers as necessary to fit the program.

2.4.1 WRITE TO THE SCREEN - OS-65D VERSION 2.0

The following routine PRINTs characters to any desired screen location. Where several lines containing PRINT statements are involved, each consecutive line of printed data begins where the last line stops.

```
10 REM--DIRECT WRITE TO SCREEN--
11 REM--STARTING VIDEO ADDRESS=D300--
12 REM--ANY ADDRESS FROM D000 TO THE END--
13 REM--OF VIDEO RAM MAY BE USED--
20 POKE 11860,0:REM--LOW BYTE OF D300--
21 REM--INTO MEMORY OUTPUT POINTER--
30 POKE 11861,211:REM--HI BYTE OF D300--
31 REM--INTO MEMORY OUTPUT POINTER--
40 D = PEEK (8708)
60 PRINT "WHATEVER YOU WANT CAN BE DISPLAYED"
70 PRINT "HERE: STATEMENTS, CHARACTER STRINGS"
80 PRINT "ETC."
90 POKE 8708,D:REM--RESTORES OUTPUT FLAG--
91 REM--TO VALUE IT HAD BEFORE THIS ROUTINE--
```

NOTE: If a CONTROL-C is typed while this routine is running, the contents of memory location 8708 may not be restored. Difficulties may be encountered with scrolling. To restore the system, type POKE 8708,2 (return) in the immediate mode.

2.4.2 MEMORY TRANSFERS - OS-65D VERSION 2.0

The contents of the video RAM may be loaded to disk, or the contents of the disk can be loaded back to RAM through the following general-purpose routine:

```
10 REM--GENERAL DISK SUBROUTINE--
20 A = 11290:REM--BIN TRACK FILE PARAMETER--
30 POKE A,T:REM--SEE NOTE 1--
40 POKE A+1,1:REM--SECTOR NUMBER - ALWAYS 1--
50 POKE A+3,0:REM--START VIDEO - LOW BYTE--
60 POKE A+4,208:REM--START VIDEO - HI BYTE--
70 POKE A+5,SIZE:REM--SECTOR LENGTH - SEE NOTE 2--
80 POKE 11812,FUNCT:REM--USR DISPATCHER - NOTE 3--
90 X=USR (X):REM--CALLS UP SUBROUTINE--
100 RETURN
```

NOTES:
1. T = Free track number. Track numbers 0 - 9 must be prefaced with a 0, i.e. 01, 02, 03...09.
2. Video memory size for 540 = 8 pages. Statement 70 is POKE A+5,8
   Video memory size for 440 and 600 = 4. pages. Statement 70 is POKE A+5,4
3. Following functions are selected by POKEing the appropriate decimal value for FUNCT TO 11860.

FUNCTion	HEX	DEC
Output to Disk A	31	49
Input from Disk A	2A	42
Output to Disk B	D1	209
Input from Disk B	D7	215

4. To load video memory to disk, routine line numbers must be sequenced after video program.

2.4.3 WRITE TO THE SCREEN - OS-65D VERSION 3.0

The following routine may be used to PRINT a line starting at some video screen location rather than at the bottom. The starting point on the screen is specified as a hex address (see video memory maps as needed).

```
10 DISK! "MEM, D000":REM---OR UP TO---
20 REM--HIGHEST VIDEO ADDRESS---
30 PRINT #4, "------LINE------"
```

2.4.4 TRANSFER SCREEN TO DISK - OS-65D VERSION 3.0

The following routine may be used within a program to store the contents of the video memory on disk:

```
10 DISK! "SAVE T,1 = D000/8":REM--SEE NOTES--
```

NOTES:

1. T = Free track number. Track numbers from 0 to 9 must be prefaced by 0, i.e. 01, 02, 03...09.
2. 1 = starting sector (always 1).
3. D000/8 = Starting address, 8 pages. For 440 or 600 boards, use D000/4
4. Alter routine line numbers as necessary.

2.4.5 TRANSFER DISK TO SCREEN - OS-65D VERSION 3.0

The following routine may be used within a program to transfer the contents of a disk track to the video memory.

```
10 DISK! "CALL D000 = T,1"
20 REM--SEE NOTES -"TRANSFER SCREEN TO DISK"--
```

2.5 BASIC ERROR CODES - CHARACTER GRAPHICS SYSTEMS

Table 2-1 lists a series of two-character BASIC error messages which are incorporated into character graphics systems. These represent the same error conditions as the standard error codes. The form of the codes differs in that the second letter has been replaced by a graphics character.

TABLE 2-1. BASIC ERROR CODES.

CODE		DEFINITION
DD	D	Double Dimension: Variable dimensioned twice. Remember subscripted variables default to dimension 10.
FC	F	Function Call error: Parameter passed to function out of range.
ID	I	Illegal Direct: Input or DEFIN statements can not be used in direct mode.
NF	N	NEXT without FOR:
OD	O	Out of Data: More reads than DATA
OM	O	Out of Memory: Program too big or too many GOSUBs, FOR NEXT loops or variables
OV	O	Overflow: Result of calculation too large for BASIC.
SN	S	Snytax error: Typo, etc.
RG	R	RETURN without GOSUB
US	U	Undefined Statement: Attempt to jump to non-existent line number
/Ø	/	Division by Zero
CN	C	Continue errors: attempt to inappropriately continue from BREAK or STOP
LS	L	Long String: String longer than 255 characters
OS	O	Out of String Space: Same as OM
ST	S	String Temporaries: String expression too complex.
TM	T	Type Mismatch: String variable mismatched to numeric variable
UF	U	Undefined Function

SECTION 3 PROGRAMMED KEY FUNCIONS

3.1 THEORY

The OSI model 542 polled keyboard and model 600 board contain a firmware-scanned switch matrix which is outwardly similar to a standard ASCII keyboard. Figures 3-1 and 3-2 show the layouts of the switch matrices and key faces. The I/O port for the polled keyboard resides at memory location DF00(hex) or 57088(dec).

In operation, the polling routine successively addresses each row of key switches R0 - R7. Between these row scans, the routine checks the columns C0 - C7 for closed key switches. If a key closure is detected, the polling routine supplies the CPU with the ASCII code corresponding to the face of the key pressed. Each of the rows is addressed in turn, thus all key switches are scanned rapidly.

The BASIC statements used for programming special keyboard functions are POKE 57088, (row address) and IF PEEK (57088) = (column address). After RUN is entered, these statements assume control of the key board since the normal polling routine is disabled (except where INPUT statements are encountered). In essence, the POKE statement turns on a row of keys, and the PEEK statement monitors the columns for a key closure. Upon detection of a closure, the PEEK statement can then trasfer control to subroutines, GOTO statements, etc. This permits the function of each key can be| software-defined for implementation of passwords, gaming controls, etc.

3.2 CONTROL-C DISABLE

The polling routine is supplemented by a second routine which monitors the CONTROL and C keys while a program is running. During this period, the full-keyboard polling routine is disabled, and the latter routine allows the user to exit a running program when CONTROL-C is entered. The CONTROL-C routine must also be disabled to allow programming special keyboard functions. The address and data POKEd depends upon the system configuration and whether or not a disk system is used. The necessary data for disabling CONTROL-C on these systems is contained in Table 3-1.

	BASIC-IN-ROM		OS-65D V2.0, 2.2 6-DIGIT		OS-65D V2.0, 2.2 9-DIGIT		OS-65D V3.0 9-DIGIT	
	POKE 530, X		POKE 9167, X		POKE 2073, X		POKE 8406, X	
	OFF	ON	OFF	ON	OFF	ON	OFF	ON
VALUE OF X	1	0	96	169	96	76	0	3

TABLE 3-1. CONTROL-C DISABLE FOR BASIC-IN-ROM AND DISK SYSTEMS - 600 OR 540/542 CONFIGURATIONS.

3.3 PROGRAMMING DIFFERENCES - 542 AND 600 BOARDS

The hardware associated with the polling routine and keyboard differs slightly between the 542 and 600 boards. These differences require slight alterations of the programs which provide specialized key functions.

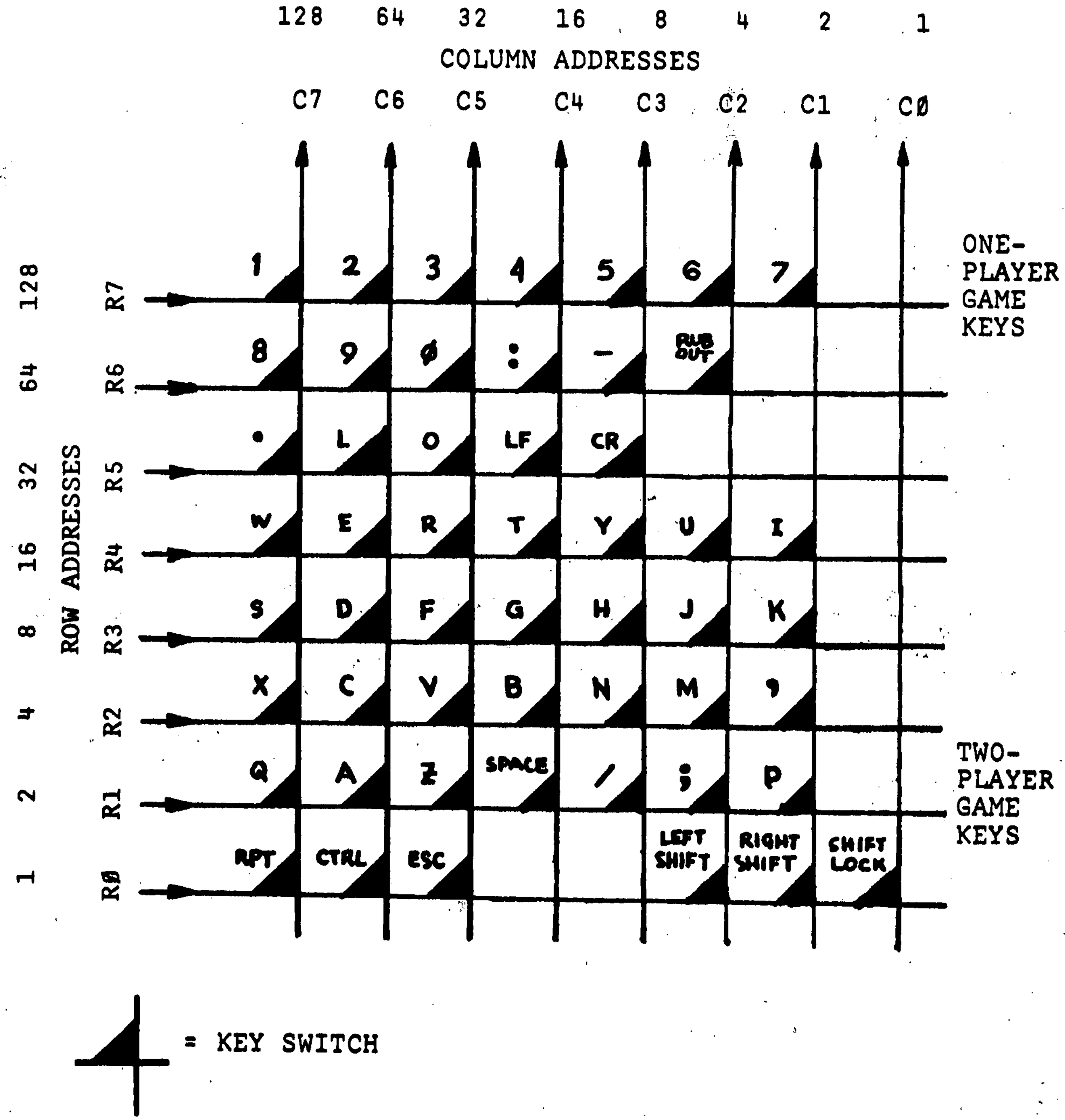
These differences are reflected in the row and column addresses in Figures 3-1 and 3-2. With a 542 board, these addresses range from 1 to 128 as powers of two. With the 600 board, these addresses are the inverses of those encountered with the 542 board. POKE statements which turn on the key rows must contain row addresses which correspond to the computer system being programmed.

Statements which implement multiple-key functions also vary slightly. With the 542, a PEEK statement takes the form:

```
(line) IF PEEK (57088) = (CA OR CB OR...ETC.) THEN (line)
```

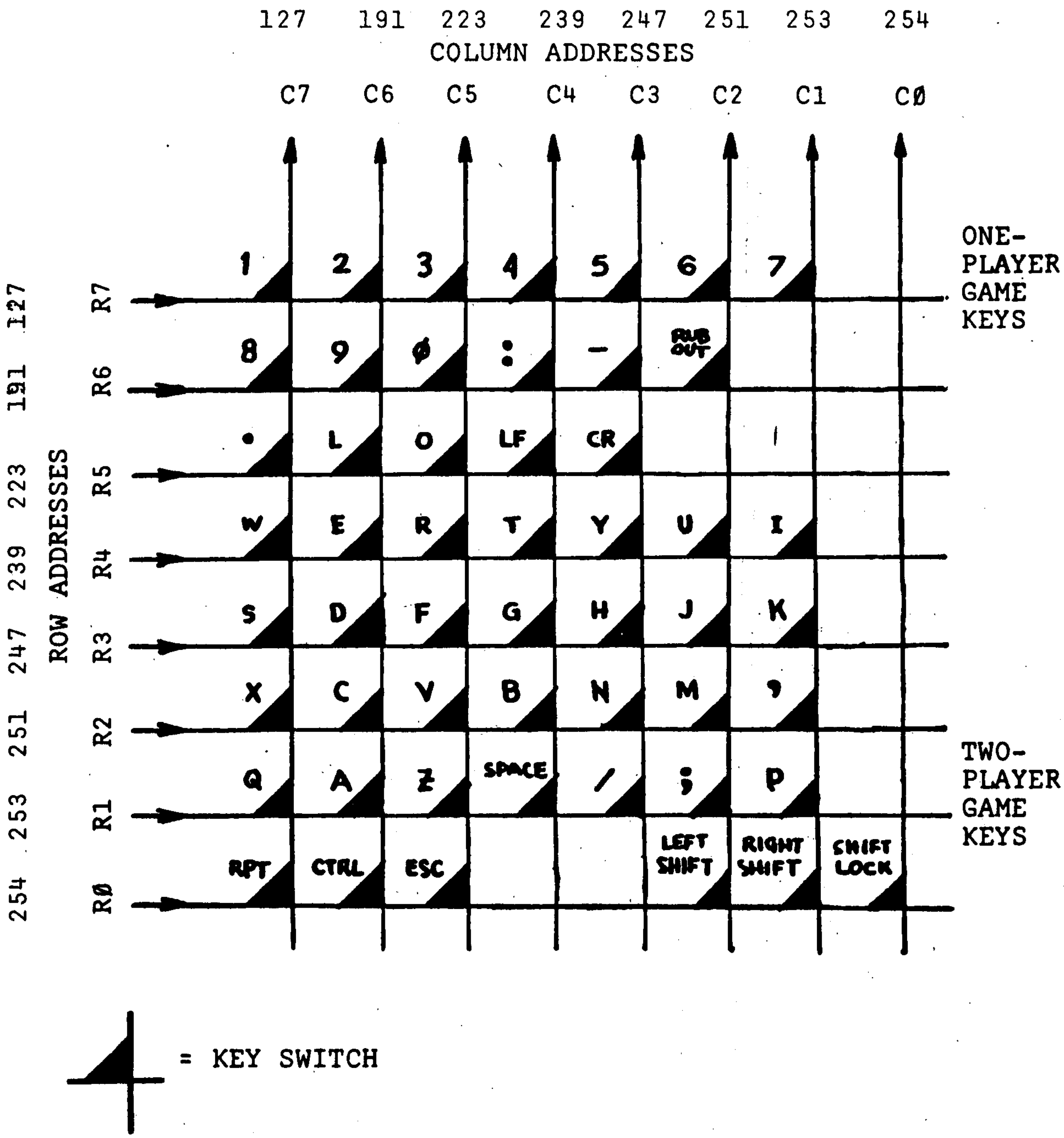
CA and CB are addresses of the columns being monitored. Boolean OR expressions join the column addresses and parentheses bound the column addresses. A similar statement for a 600-based system would be:

OSGSD 3.3 KEYBOARD ROUTINE AT $3590



NOTES: 1. Standard 53-key layout except:
"HERE IS" deleted, "RUB OUT" at "HERE IS" location,
"SHIFT LOCK" at "RUB OUT" location.
2. "LEFT SHIFT" and "RIGHT SHIFT" separately decoded.

FIGURE 3-1. SWITCH MATRIX - 542 POLLED KEYBOARD.



NOTES: 1. Standard 53-key layout except:
"HERE IS" deleted, "RUB OUT" at "HERE IS" location,
"SHIFT LOCK" at "RUB OUT" location.
2. "LEFT SHIFT" and "RIGHT SHIFT" separately decoded.

FIGURE 3-2. SWITCH MATRIX - 600 POLLED KEYBOARD.

```
(line) IF PEEK (57088) = (CA AND CB AND...ETC.) THEN (line)
```

Note that the OR expressions have been changed to AND, but that the statements are otherwise identical. Following are two examples for programs which utilize programmed key functions.

3.4 DEMO PROGRAMS - KEYBOARD FUNCTIONS

```
5 REM---EXAMPLE NO. 1 ----
10 REM--DETECTS "G" SWITCH CLOSURE - 542 BOARD--
20 POKE 530,1 : REM--CONTROL-C DISABLE - SEE NOTE 1--
30 K=57088
40 POKE K,8 : REM--FOR 600 SEE NOTE 2--
50 IF PEEK(K) = 16 THEN 100 : REM--FOR 600 SEE NOTE 3--
60 GOTO 50 : REM--GO BACK AND PEEK AGAIN--
100 PRINT "G-KEY PRESSED" : REM--SEE NOTE 4--
110 POKE 530,0 : REM--RESTORES CONTROL C-- : REM--NOTE 1--
```

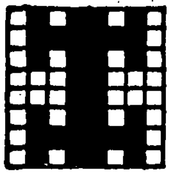
NOTES:
1. For disk system, POKE proper CONTROL-C disable and enable (see Table 3-1).
2. For 600 board, POKE K,247
3. For 600 board, IF PEEK (K) = 239 THEN 100
4. This line may transfer control to other lines.

```
5  REM----EXAMPLE NO. 2----
10 REM----MULTI-KEY DEMO----
20 REM--CLEAR SCREEN BY SCROLL-UP--
21 FOR X=0 TO 35 : PRINT : NEXT X
25 PRINT"INSTRUCTIONS: 1=FWD  2=REV  3=FIRE  7=EXIT"
30 REM---TARGETS---
31 FOR X=54976 TO 55000 STEP 2 : POKE X,246 : NEXT X
50 K=57088 : A=54288 : POKE 530,1 : POKE K,128
60 REM---ENTERPRISE IN TWO PARTS---
61 POKE A,11 : POKE A+1,12 : FOR X=0 TO 50 : NEXT X
70 IF PEEK (K)=128 THEN 100 : REM--128 FOR "1" KEY--
75 IF PEEK (K)=64 THEN 200 : REM--64 FOR "2" KEY--
80 IF PEEK (K)=32 THEN 300 : REM--32 FOR "3" KEY--
85 IF PEEK (K)=2 THEN 400 : REM--2 FOR "7" KEY--
90 GOTO 70 : REM--GO BACK & PEEK AGAIN--
100 POKE A,32:POKE A+1,32:A=A-1:GOTO 60
200 POKE A,32:POKE A+1,32:A=A+1:GOTO 60
300 FOR R=A+63 TO 55000 STEP 63 : POKE R,189
310 FOR X=0 TO 10 : NEXT X : POKE R,32 : NEXT R
320 GOTO 80
400 POKE 530,0 : REM--RESTORE CONTROL C-- : END
```

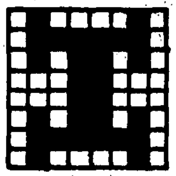
NOTES:
1. For disk systems, POKE proper CONTROL-C disable and enable (see Table 3-1).
2. To run this program on a 600-based system, make the following line number changes:

```
31 FOR X=54051 TO 54071 STEP 2 : POKE X,246 : NEXT X
50 K=57088 : A=53549 : POKE 530,1 : POKE K,127
70 IF PEEK (K)=127 THEN 100
75 IF PEEK (K)=191 THEN 200
80 IF PEEK (K)=223 THEN 300
90 IF PEEK (K)=253 THEN 400
300 FOR R=A+31 TO 54071 STEP 31 : POKE R,189
```

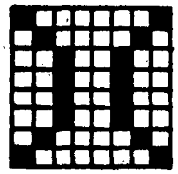
D549



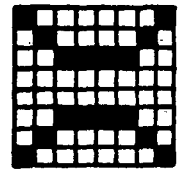
0 $0



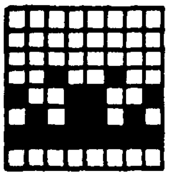
1 $1



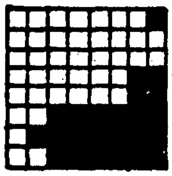
2 $2



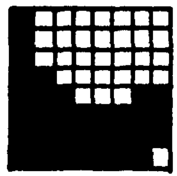
3 $3



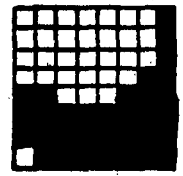
4 $4



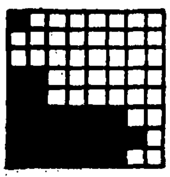
5 $5



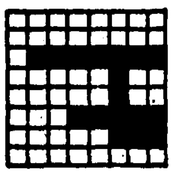
6 $6



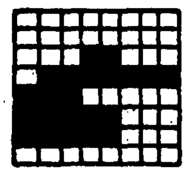
7 $7



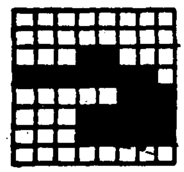
8 $8



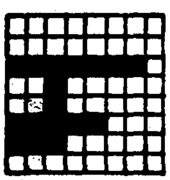
9 $9



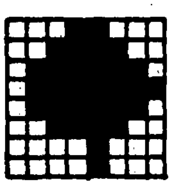
10 $A



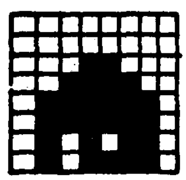
11 $B



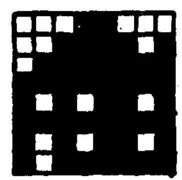
12 $C



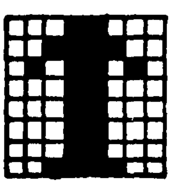
13 $D



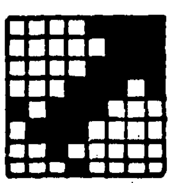
14 $E



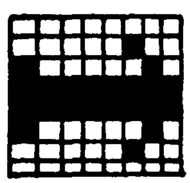
15 $F



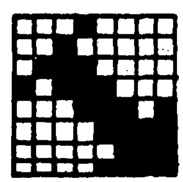
16 $10



17 $11

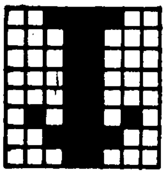


18 $12

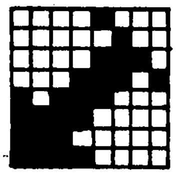


19 $13

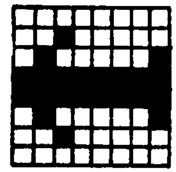
TABLE 1-1. OSI GRAPHICS CHARACTERS



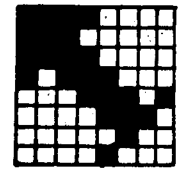
20 $14



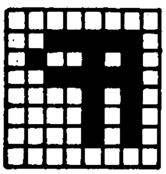
21 $15



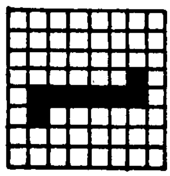
22 $16



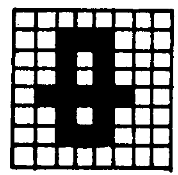
23 $17



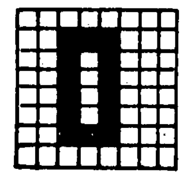
24 $18



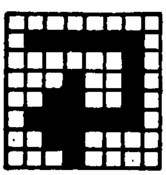
25 $19



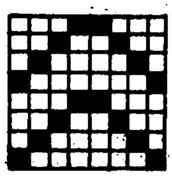
26 $1A



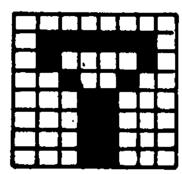
27 $1B



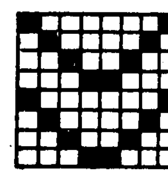
28 $1C



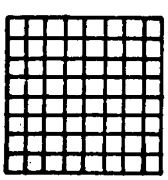
29 $1D



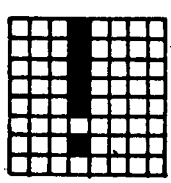
30 $1E



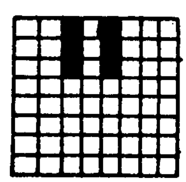
31 $1F



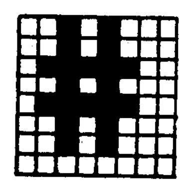
32 $20



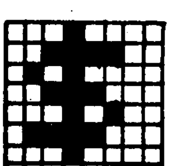
33 $21



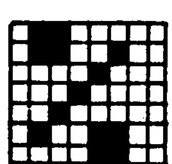
34 $22



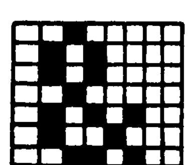
35 $23



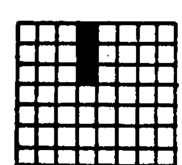
36 $24



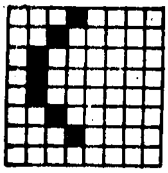
37 $25



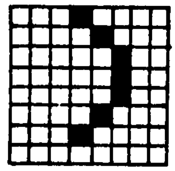
38 $26



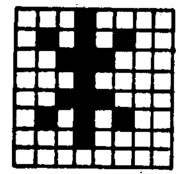
39 $27



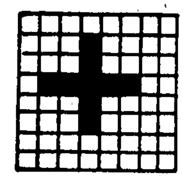
40 $28



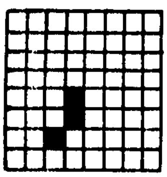
41 $29



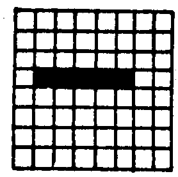
42 $2A



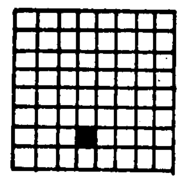
43 $2B



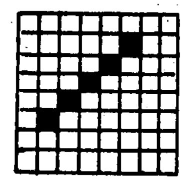
44 $2C



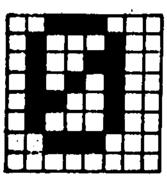
45 $2D



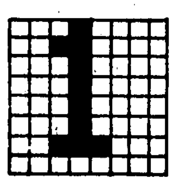
46 $2E



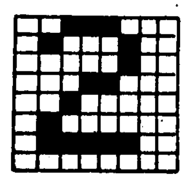
47 $2F



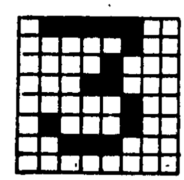
48 $30



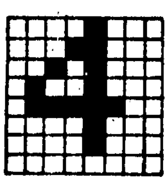
49 $31



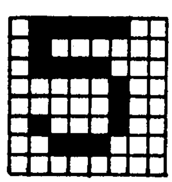
50 $32



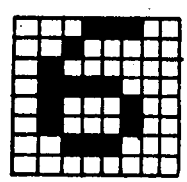
51 $33



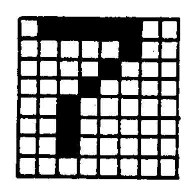
52 $34



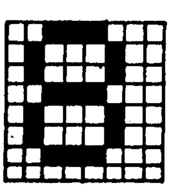
53 $35



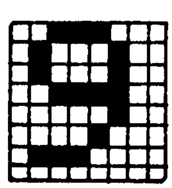
54 $36



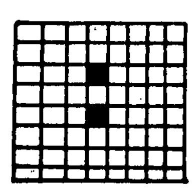
55 $37



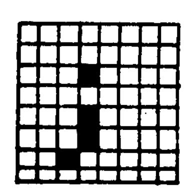
56 $38



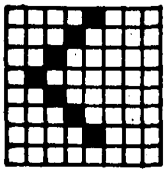
57 $39



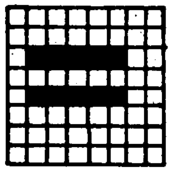
58 $3A



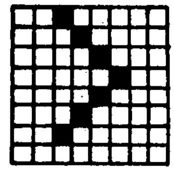
59 $3B



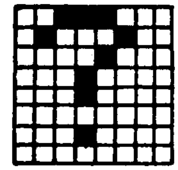
60 $3C



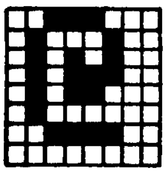
61 $3D



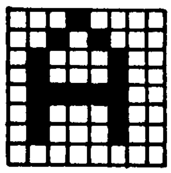
62 $3E



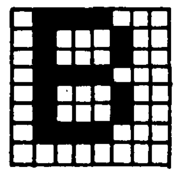
63 $3F



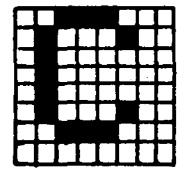
64 $40



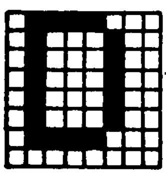
65 $41



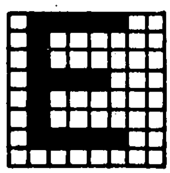
66 $42



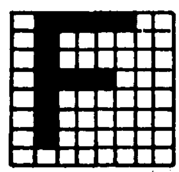
67 $43



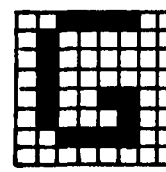
68 $44



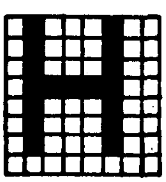
69 $45



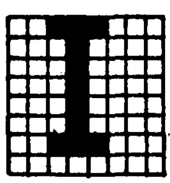
70 $46



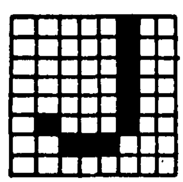
71 $47



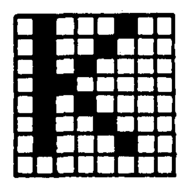
72 $48



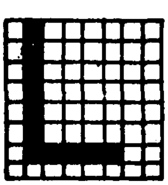
73 $49



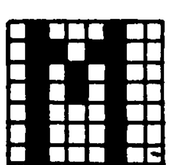
74 $4A



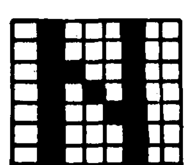
75 $4B



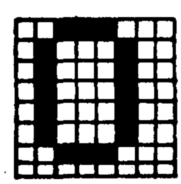
76 $4C



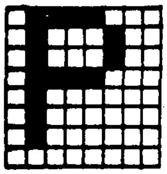
77 $4D



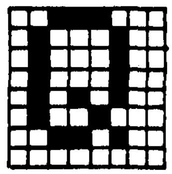
78 $4E



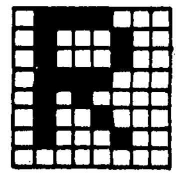
79 $4F



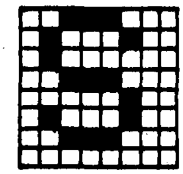
80 $50



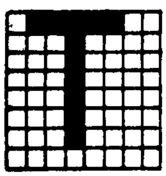
81 $51



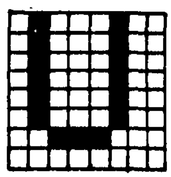
82 $52



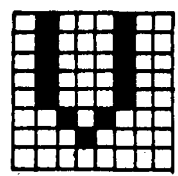
83 $53



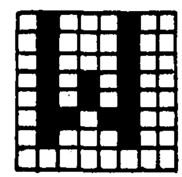
84 $54



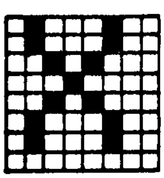
85 $55



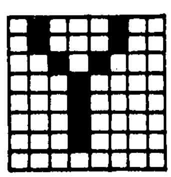
86 $56



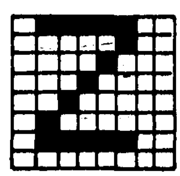
87 $57



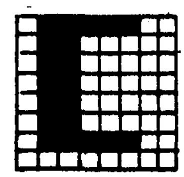
88 $58



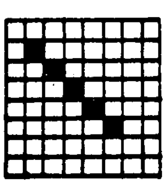
89 $59



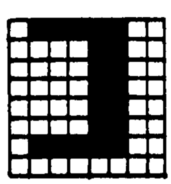
90 $5A



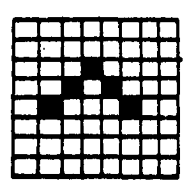
91 $5B



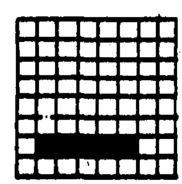
92 $5C



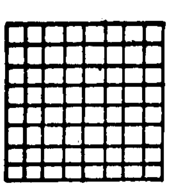
93 $5D



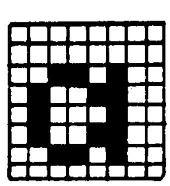
94 $5E



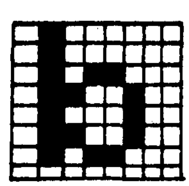
95 $5F



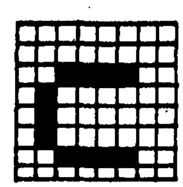
96 $60



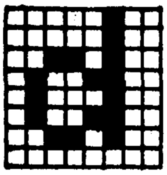
97 $61



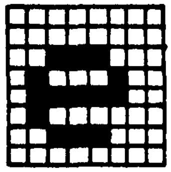
98 $62



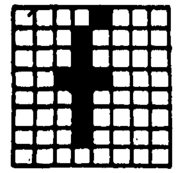
99 $63



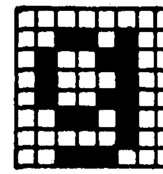
100 $64



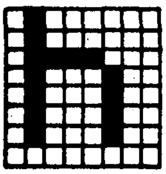
101 $65



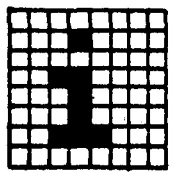
102 $66



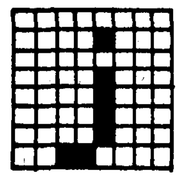
103 $67



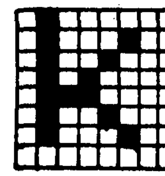
104 $68



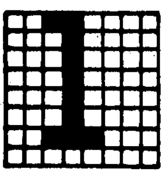
105 $69



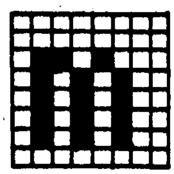
106 $6A



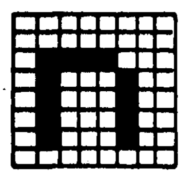
107 $6B



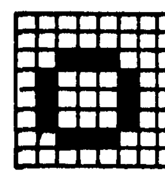
108 $6C



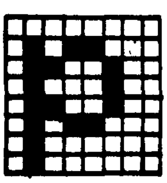
109 $6D



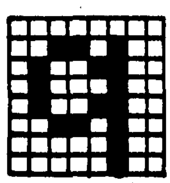
110 $6E



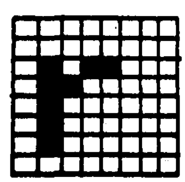
111 $6F



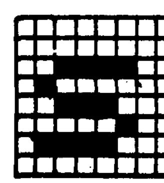
112 $70



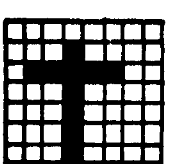
113 $71



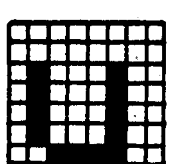
114 $72



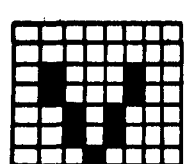
115 $73



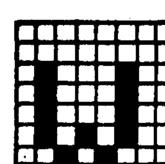
116 $74



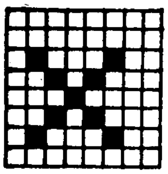
117 $75



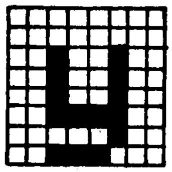
118 $76



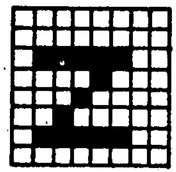
119 $77



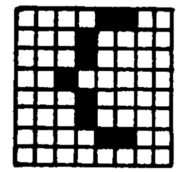
120 $78



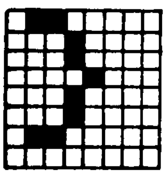
121 $79



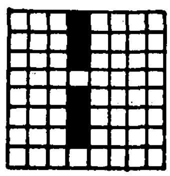
122 $7A



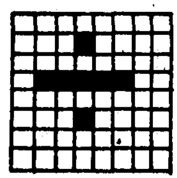
123 $7B



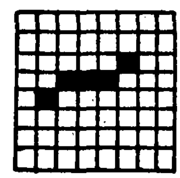
124 $7C



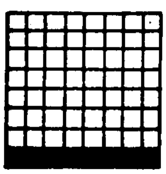
125 $7D



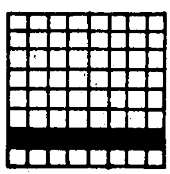
126 $7E



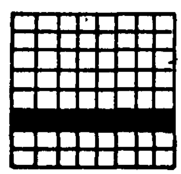
127 $7F



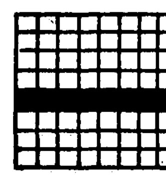
128 $80



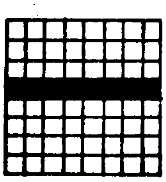
129 $81



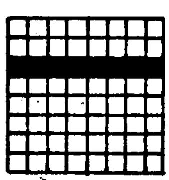
130 $82



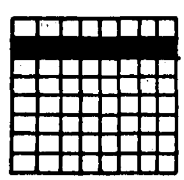
131 $83



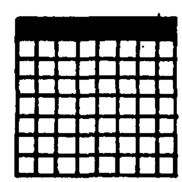
132 $84



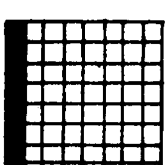
133 $85



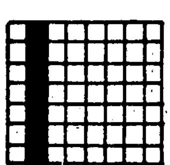
134 $86



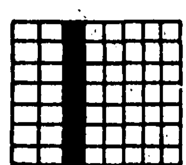
135 $87



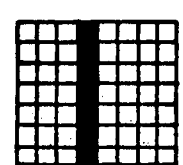
136 $88



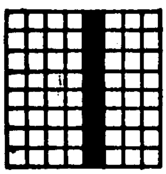
137 $89

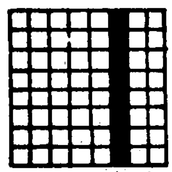


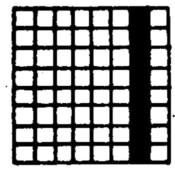
138 $8A

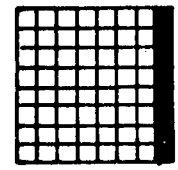


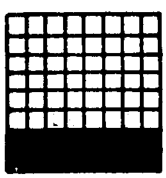
139 $8B

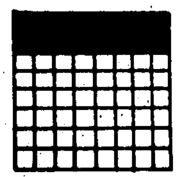

140 $8C

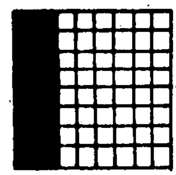

141 $8D

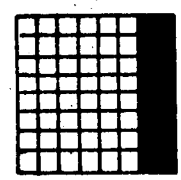

142 $8E

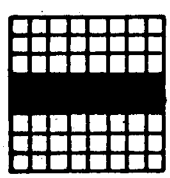

143 $8F

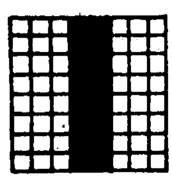

144 $90

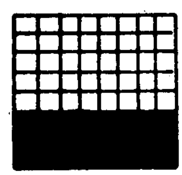

145 $91

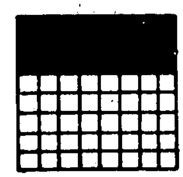

146 $92

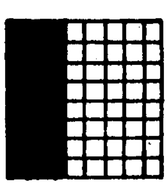

147 $93

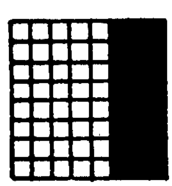

148 $94

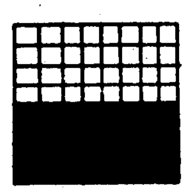

149 $95

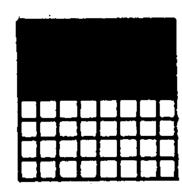

150 $96

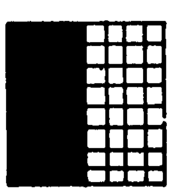

151 $97

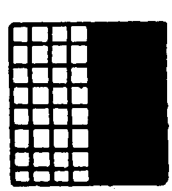

152 $98

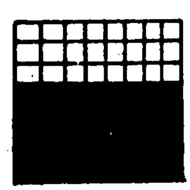

153 $99

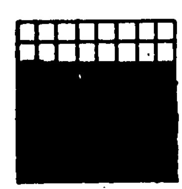

154 $9A

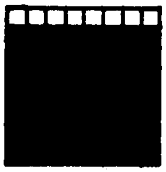

155 $9B


156 $9C

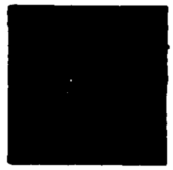

157 $9D


158 $9E

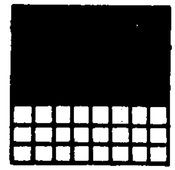

159 $9F



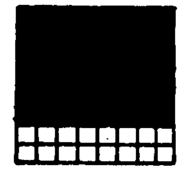
160 $A0



161 $A1

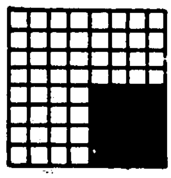


162 $A2

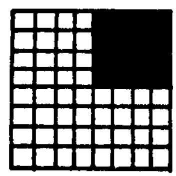


163 $A3

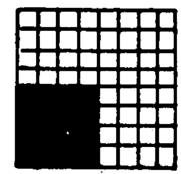
164 $A4



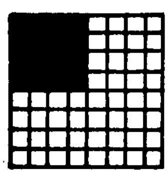
165 $A5



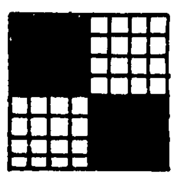
166 $A6



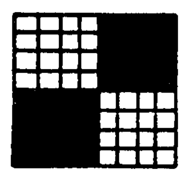
167 $A7



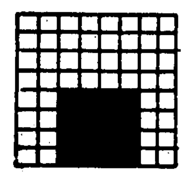
168 $A8



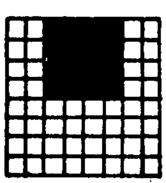
169 $A9



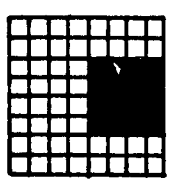
170 $AA



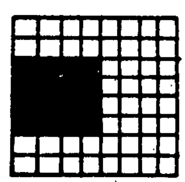
171 $AB



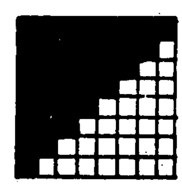
172 $AC



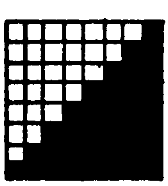
173 $AD



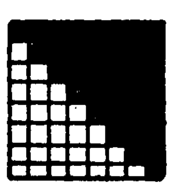
174 $AE



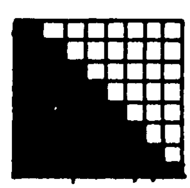
175 $AF



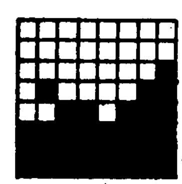
176 $B0



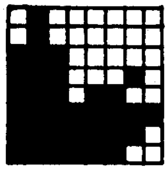
177 $B1



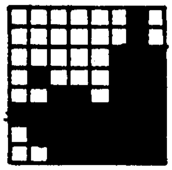
178 $B2



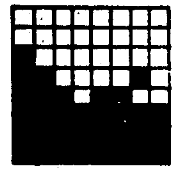
179 $B3



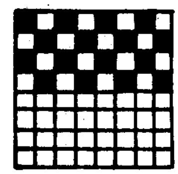
18Ø $B4



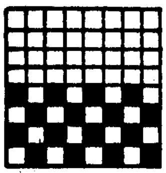
181 $B



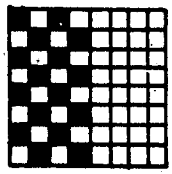
182 $B6



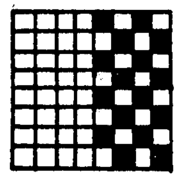
183 $B7



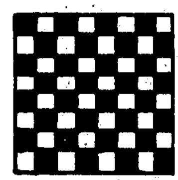
184 $B8



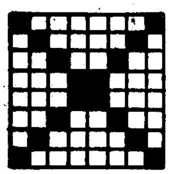
185 $B9



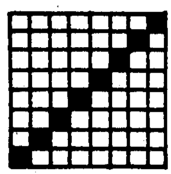
186 $BA



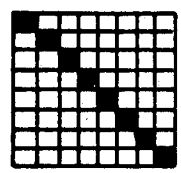
187 $BB



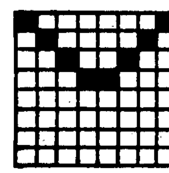
188 $BC



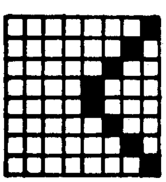
189 $BD



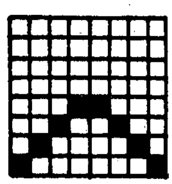
19Ø $BE



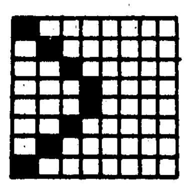
191 $BF



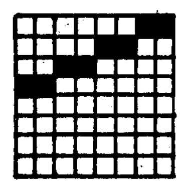
192 $CØ



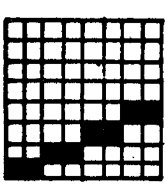
193 $C1



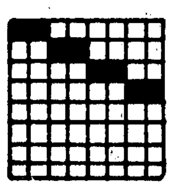
194 $C2



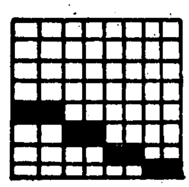
195 $C3



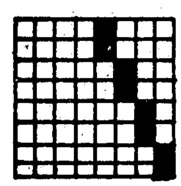
196 $C4



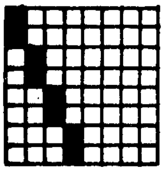
197 $C5



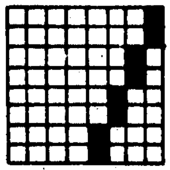
198 $C6



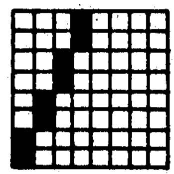
199 $C7



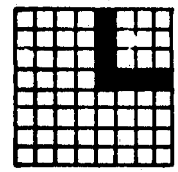
2ØØ $C8



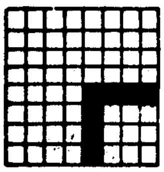
2Ø1 $C9



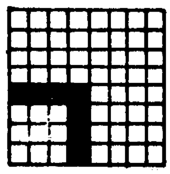
2Ø2 $CA



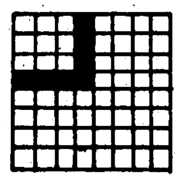
2Ø3 $CB



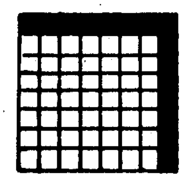
2Ø4 $CC



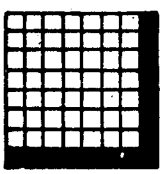
2Ø5 $CD



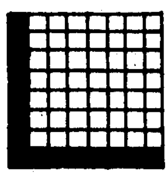
2Ø6 $CE



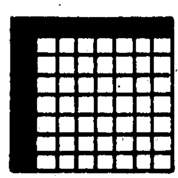
2Ø7 $CF



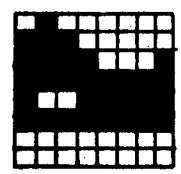
2Ø8 $DØ



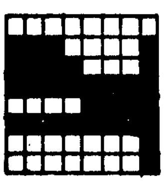
2Ø9 $D1



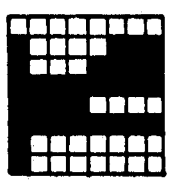
21Ø $D2



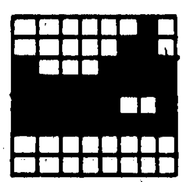
211 $D3



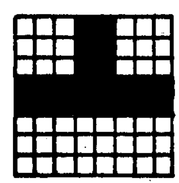
212 $D4



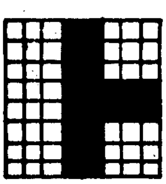
213 $D5



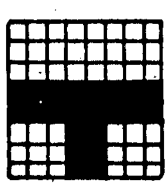
214 $D6



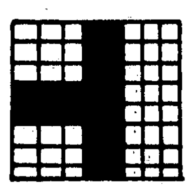
215 $D7



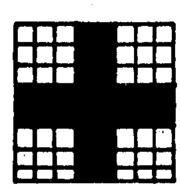
216 $D8



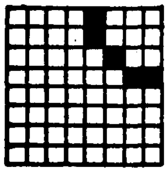
217 $D9

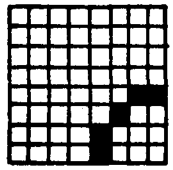


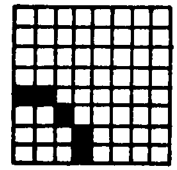
218 $DA

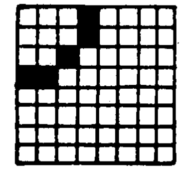


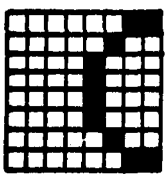
219 $DB

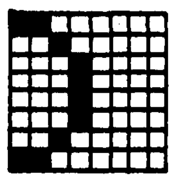

22Ø $DC

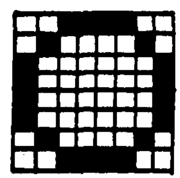

221 $DD

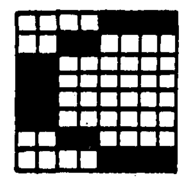

222 $DE

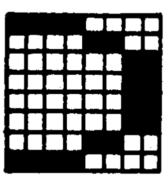

223 $DF

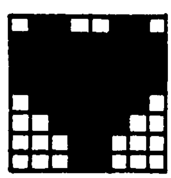

224 $EØ

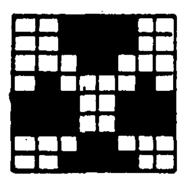

225 $E1

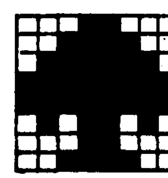

226 $E2

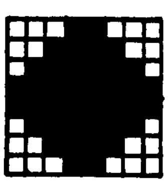

227 $E3

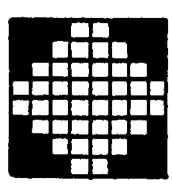

228 $E4

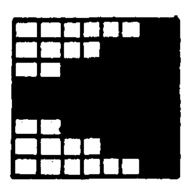

229 $E5

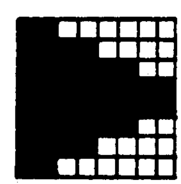

23Ø $E6

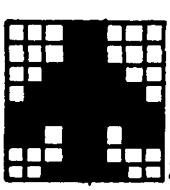

231 $E7

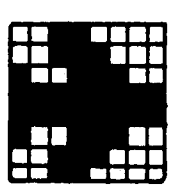

232 $E8

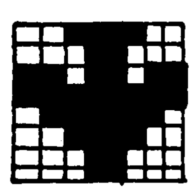

233 $E9

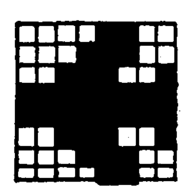

234 $EA

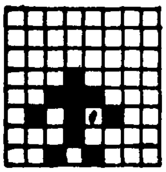

235 $EB


236 $EC

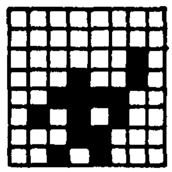

237 $ED


238 $EE

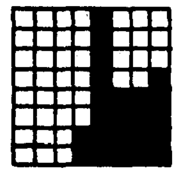

239 $EF



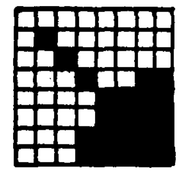
24Ø $FØ



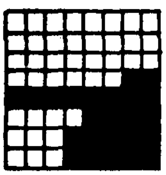
241 $F1



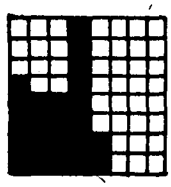
242 $F2



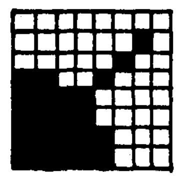
243 $F3



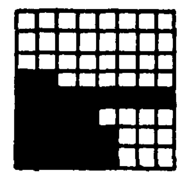
244 $F4



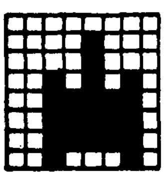
245 $F5



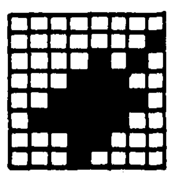
246 $F6



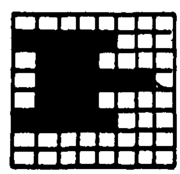
247 $F7



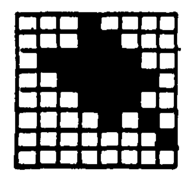
248 $F8



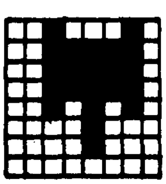
249 $F9



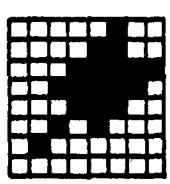
25Ø $FA



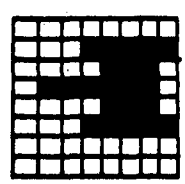
251 $FB



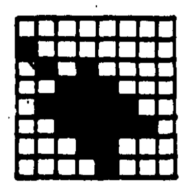
252 $FC



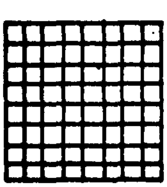
253 $FD



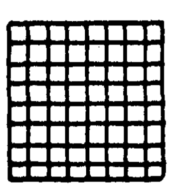
254 $FE



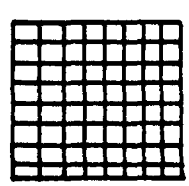
255 $FF



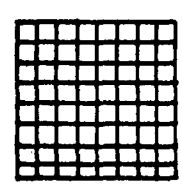
___ $___



___ $___



___ $___



___ $___

FIGURE 1-1. VIDEO MEMORY MAP - 540 IN 32x64 FORMAT.

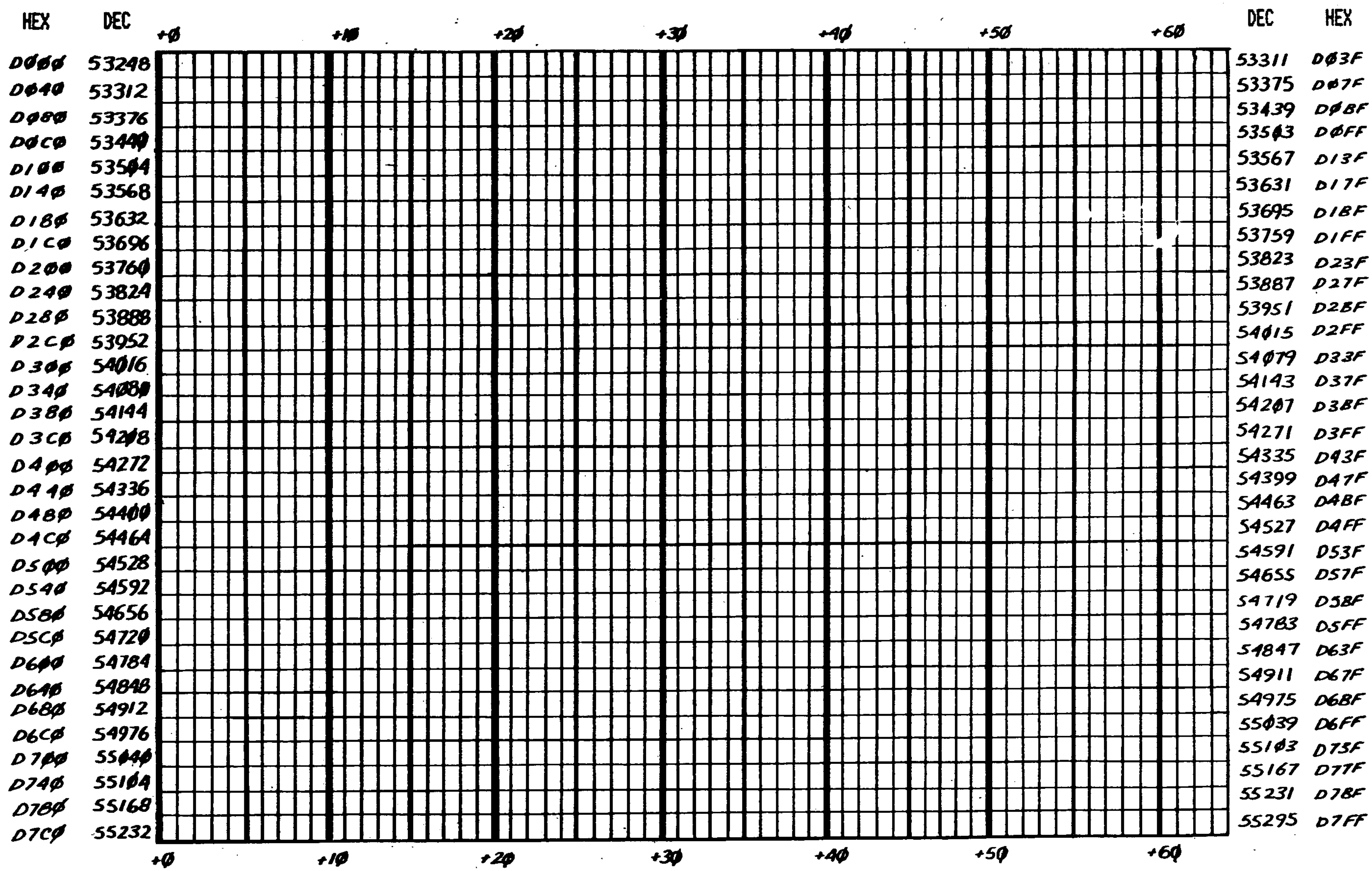


FIGURE 1-2. VIDEO MEMORY MAP - 540 IN 32x32 FORMAT.

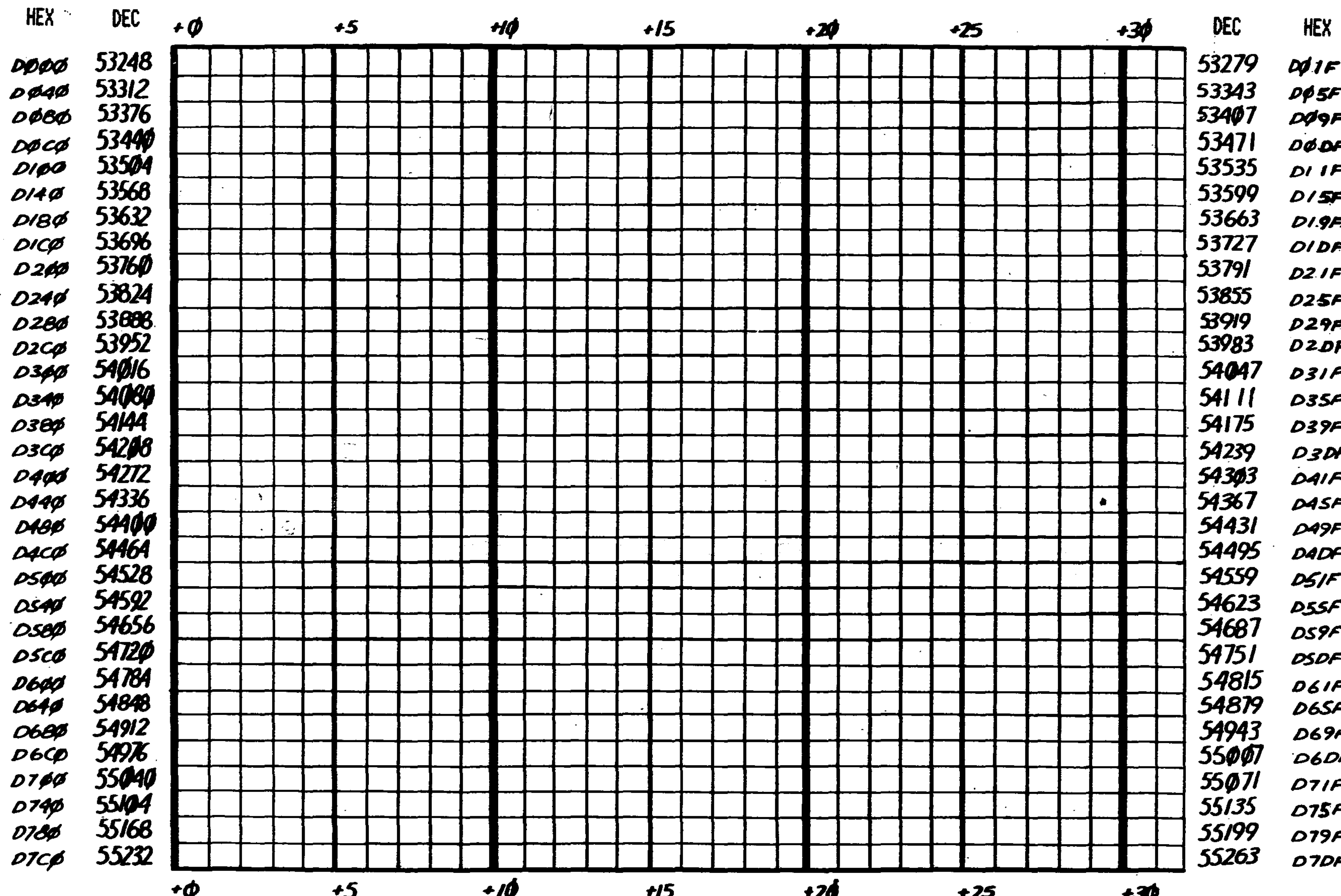


FIGURE 1-3. VIDEO MEMORY MAP - 600 IN 25x25 FORMAT.

+Ø +5 +1Ø +15 +2Ø

HEX	DEC	DEC	HEX
DØ83	53379	534Ø3	DØ9B
DØA3	53411	53435	DØBB
DØC3	53443	53467	DØDB
DØE3	53475	53499	DØFB
D1Ø3	535Ø7	53531	D11B
D123	53539	53563	D13B
D143	53571	53595	D15B
D163	536Ø3	53627	D17B
D183	53635	53659	D19B
D1A3	53667	53691	D1BB
D1C3	53699	53723	D1DB
D1E3	53731	53755	D1FB
D2Ø3	53763	53787	D21B
D223	53795	53819	D23B
D243	53827	53851	D25B
D263	53859	53883	D27B
D283	53891	53915	D29B
D2A3	53923	53947	D2BB
D2C3	53955	53979	D2DB
D2E3	53987	54Ø11	D2FB
D3Ø3	54Ø19	54Ø43	D31B
D323	54Ø51	54Ø75	D33B
D343	54Ø83	541Ø7	D35B
D363	54115	54139	D37B
D383	54147	54171	D39B

+Ø +5 +1Ø +15 +2Ø

The OSI 48 Line BUS

offers the broadest line of BUS compatible microcomputer boards. This line includes several new and exciting products which are not available anywhere else, such as a three processor CPU board, dual port memories and a multiprocessing CPU expander.

has delivered approximately 100,000 boards based on our 48 line BUS and is now delivering thousands per week in 17 models of computers and dozens of accessories.

BUS design incorporates high band width, high density and mass production technology to achieve a truly remarkable performance to cost ratio.

Here is just a sampling of the many OSI 48 BUS compatible boards available for the systems user, prototyper, OEM user and experimenter.

Product Description	Special Features	Power Supply Voltages Req'd	Board & Doc Part #	Board & Doc Price	Assembled Product Part #	Assembled Product Price
CPU						
• Challenger II CPU BASIC-in-ROM 6502 based CPU with serial I/O 4K RAM, machine code monitor	• Can use four 2716 EPROMS instead of BASIC or can be configured for disk	+5/−9	500	39.00	C2-0	298.00
• Challenger III CPU has 6502A, 6800 and Z80 micros, RS-232 serial port, machine code monitor	• 1 megabyte memory manager, software programmable vectors	+5/−9	510	NA	C3-0	490.00
• 560Z multi-processing CPU expander runs PDP-8, Z80 and 8080 code	• Runs concurrently with another OSI CPU	+5/−9	560Z	125.00	NA	NA
RAM						
• 16K static RAM (Ultra low power)	• 215NS access time automatic power down standby mode	+5/+12/−9	520	35.00	CM-3	498.00
• 8K static RAM (low cost)	• Expandable to 16K	+5	—	—	CM-7	198.00
• 16K static RAM (low cost)	• Can be expanded to dual port operation	+5	525	35.00	CM-8	339.00
• 24K static RAM (high density)	• 20 address bits	+5	527	35.00	CM-9	NA
• 4K static RAM (2102 based)	• Can be populated for 4K by 12 bits	+5	420	35.00	CM-2	125.00
• 16K dynamic (ultra low cost)	• Uses 4027 RAMS	+5/+12/−9	530	NA	CM-4	249.[illegible]
• 32K dynamic	• 20 address bits	+5/+12/−9	530	NA	CM-5	698.00
• 48K dynamic (high density)	• 20 address bits	+5/+12/−9	530	NA	CM-6	960.00
EPROM Boards						
• 8K 6834 EPROM board	• 16 line parallel port and on board programmer	+5/−9	450	35.00	NA	NA
• 4K 1702A EPROM board	• 16 line parallel port	+5/−9	455	35.00	NA	NA
I/O Boards						
• Audio Cassette interface Kansas City standard 300 baud	• Expandable to CA-7C	+5/−9	430	35.00	CA-6C	99.00
• RS-232 port board	• Expandable to CA-7S	+5/−9	430	35.00	CA-6S	99.00
• Combination audio cassette two 8 bit DACs, one fast A/D and 8 channel input mux	• Also Features 8 parallel I/O lines	+5/−9	430	35.00	CA-7C	399.00
• Combination RS-232 two 8 bit DACs, one fast A/D and 8 channel input mux	• Also features 8 parallel I/O lines	+5/−9	430	35.00	CA-7S	399.00
• 32 by 32 character video display interface	• Keyboard input port	+5/−9	440	35.00	NA	NA
• 32 by 64 character video display interface	• Upper/lower case graphics and keyboard port	+5	540	NA	CA-11	249.00
• 16 port serial board RS-232 and/or high speed synchronous	• 75 to 19,200 baud and 250K and 500K bit rates individually strappable	+5/−9	550	35.00	CA-10X	200.00 to 900.00
• Parallel (Centronics) Line Printer Interface	• With cable	+5/−9	470	NA	CA-9	249.00
• 96 Line Remote Parallel Interface	• Interface "Front End" remotable via 16 pin ribbon cable	+5	—	—	CA-12	249.00
• Voice I/O board with Votrax* module	• Fully assembled voice output, experimental voice input	+5/−9	—	—	CA-14	525.00
DISKS						
• Single 8" floppy disk, 250 Kbytes storage	• Complete with operating system software and disk BASIC	+5/−9	470	NA	CD-1P	790.00
• Dual 8" floppy disk, 500 Kbytes storage	• Complete with operating system software and disk BASIC	+5/−9	470	NA	CD-2P	1390.00
• 74 Million byte Winchester disk and interface	• Complete with OS-65U operating system	+5/−9	—	—	CD-74	6000.00
OTHER						
• 8 slot backplane board with connectors	• Can be daisy-chained to n-slots	—	580	39.00	NA	NA
• Prototyping board	• Handles over 40 16 pin IC's	—	495	29.00	—	—
• Card Extender	• With connectors	—	498	29.00	—	—

For more information, contact your local OHIO SCIENTIFIC Dealer or the factory at (216) 562-3101

OHIO SCIENTIFIC

1333 S. Chillicothe Road • Aurora, Ohio 44202