A VISUAL DISPLAY SYSTEM SUITABLE FOR TIME SHARED USE

BY

JESSE T. QUATSE

Carnegie Institute of Technology
June 7, 1965

This work was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense (SD-146).

ACKNOWLEDGEMENTS

I wish to thank Alan J. Perlis, Allen Newell, Robert Braden, Arthur Evans, Jr., and Einar Stefferud for their many invaluable suggestions and observations during the creation of the display system. Also, I wish to thank Andrew Kleinschnitz of Philco for his important contributions to the development and implementation of the system.

A VISUAL DISPLAY SYSTEM SUITABLE FOR TIME SHARED USE

SUMMARY

A visual display facility is being integrated with the time-shared G-21 multi-processor system at Carnegie Institute of Technology. Initially, three consoles will time-share an 8K regeneration memory which is addressable as main memory by the G-21 processors. More consoles can be added as the need arises.

This paper documents the visual display system design. The primary design objectives were to develop and exploit a representation of displayed information which was suitable, economically and functionally, to time-shared use . The adopted representation permits a set of common editing operations to be completely controlled by relatively inexpensive console circuits. During these operations, no processor intervention is required. When processor intervention is required, the dual purpose of the memory module, as both regeneration memory and processor memory, permits a close partnership between the processors and the human console user.

TABLE OF CONTENTS

INTRODUCTION	1
The Intent	1
System Features	2
THE SYSTEM COMPONENTS	6
The G-21 System	6
The Memory	7
The Display Equipment	9
The Console Components	10
DATA REPRESENTATION IN THE REGENERATION MEMORY	15
Memory Organization	15
The System Region	16
The Display Region	19
DATA CONTROL BY THE CONSOLE OPERATOR	27
The State Switches	27
Output	29
Input	32
REFERENCES	45
NOTE	46
APPENDIX 1 - THE CHARACTER GENERATOR	A1-1
APPENDIX 2 - THE VECTOR GENERATOR	A2-1
APPENDIX 3 - THE FORMAT GENERATOR	A3-1
APPENDIX 4 - THE CONSOLE MAIN FRAME	A4-1
APPENDIX 5 - THE CURSOR CONTROL AND INTERRUPT PANEL	A5-1
APPENDIX 6 - THE TABLET	A6-1
APPENDIX 7 - THE KEYBOARD AND CHARACTER SET	A7-1

A VISUAL DISPLAY SYSTEM SUITABLE FOR TIME SHARED USE

INTRODUCTION

The Intent

The Computation Center at Carnegie Institute of Technology is in the process of expanding the G-21 computer system to include a high performance, modestly priced, visual display facility. The expanded system can support eight visual display consoles, although only three are being installed initially. Further development of the display facility will proceed by addition and modification. The addition of optical copying equipment and the modification of the vector-locating circuitry are under current consideration. If the need arises, expansion beyond eight consoles is possible.

The purpose of the expansion is to provide the Computation Center with a means of pursuing research requiring human-computer interactions at human rates. Existing low-capacity consoles, such as teletype units, will continue to serve as the inexpensive mass medium. The display equipment is intended to complement rather than supplant.

Although the equipment is specifically designed to the needs of CIT, the needs are of a general nature. Namely, a set of general purpose visual display consoles must operate on-line with an existing computer system without absorbing full computational capacity. The electronic interface to the G-21 can be adapted to any computer system utilizing multi-terminal memory access portals.

This paper serves as the single defining document of the expanded system. Sufficient detail is given for the preparation of engineering

specifications and programming manuals. Further development will be accompanied by addenda to this paper.

System Features

The central feature of the display facility is the nature of its integration with the G-21. Because of the near-capacity use of the G-21, the display equipment is required to operate autonomously "off-line" whenever possible. An optimization was necessary which minimized the cost of special display system circuitry while minimizing the equivalent cost of G-21 load. The optimization led to a block-structured regeneration memory accessible by both the G-21 processor and the display console. Within each block, data is represented in "list-block" form described later in this section.

The blocking of regeneration memory is schematized in Figure 1. The region called "system" serves as buffer memory for the console control circuitry. All console controls are channeled through the system cells so that the processor program may both monitor and initiate console operations. Thus, all operations under control of the console user are also under control of the processor program. The processor stands as a silent but omnipotent partner of the console user.

The system region occupies the first 104_{10} words of regeneration memory. The remaining $8,004_{10}$ words are called the "display" region. Starting addresses and lengths of blocks in the display region are fixed by processor program. Display region blocks, called "text blocks",

are shown in Figure 1. A display command in the head cell of each block specifies the last absolute address of the block and the console configuration appropriate to the block. Legal console configurations permit any combination of the four consoles to be viewing the display data, but at most one console to be loading information into the text block. Thus, each text block is relocatable into any absolute addresses and onto any console by updating the single head cell.

Control circuitry sequences through regeneration memory displaying text as specified by data in the display region. Upon encountering a head cell, the control circuitry establishes a new console configuration and continues accessing in sequential order. Display commands are available which shorten the cycle time, through regeneration memory, by jumping control past unused or irrelevant cells. In Figure 1, regeneration memory is effectively ($\gamma - \beta + \alpha - 150$) words long.

The processor program controls the regeneration sequence by loading head cells. Within the processor program, the structure of regeneration memory can be represented as a table of head cell addresses. Addressing through the table then eliminates unnecessary block transfers of display data. Once loaded, a block of display data need not be over-layed until memory capacity is exhausted. Resident pages may be turned by single store commands addressed through the head cell table.

A degree of console autonomy is achieved by means of the data representation within text blocks. Information entered or modified from a console is stored in a standard form called "list-block". A list-

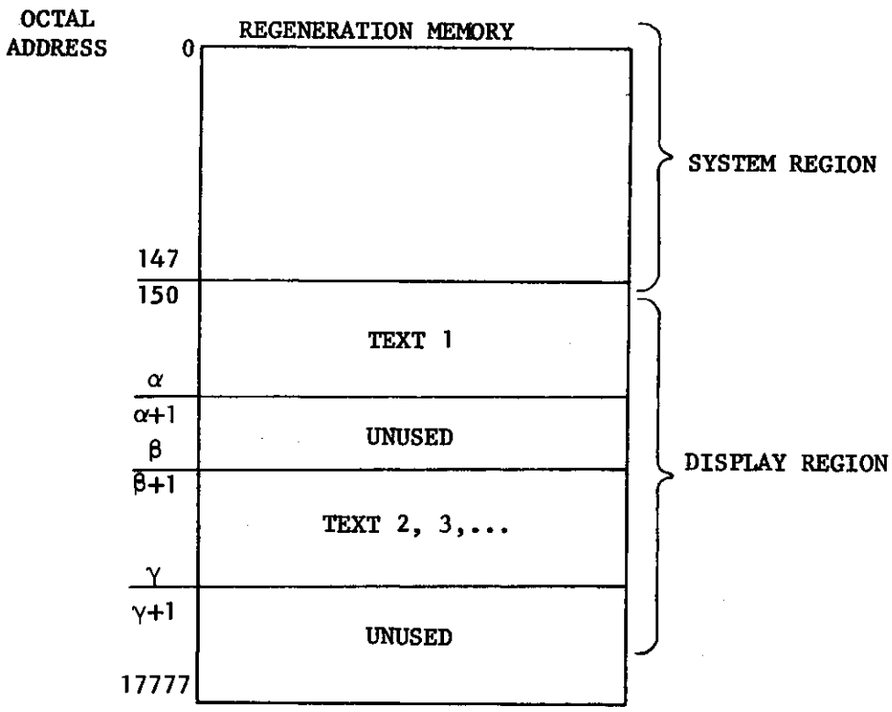


Figure 1

BLOCKING OF REGENERATION MEMORY

block is a block of contiguous cells having two list-like properties. First, the initial cell of the list-block serves the purpose of a descriptive header. Header information identifies the list-block type and initial display coordinates. The two list-block types are "Character String" and "Vector String". The character string header locates the

first character of a string on the CRT face. The string is packed in subsequent cells three characters to the 32-bit word. The vector string header locates the initial point of a figure. The next cell defines the abscissa and ordinate displacements of the next vertex of the figure. The initial point and the first vertex become end-points of a line drawn automatically by control circuitry. Subsequent cells define the remaining vertices. The figure appears on the display tube face as a series of visible or invisible lines connecting vertices in their order of list-block appearance.

The second list-like property of the list-block is its growth capability. Any list-block may be extended, from the console, by the addition of contiguous end cells. Control circuitry automatically relocates information, during the regeneration cycle, to provide additional end cells. Relocation spans only the text block so that all text blocks may be relocating simultaneously without the need for a hardware stack. By virtue of the list-block, information entered or modified from the console is represented in a standard form. No processor intervention is required to reorganize, regroup, or reduce incoming data since the list-block form is an acceptable standard.

The control circuitry for implementing block and list-block structures is not significantly more costly then that required to sequence through regeneration memory. In particular, no complex machinery such as stacks is required. The effect is a tight coupling of the human operator to the computer system by means of a mutually accessible re-

generation memory. Through tight coupling, the operator and the computer can become indistinguishable partners. Independence is achieved by the convenient representation of displayed data.

THE SYSTEM COMPONENTS

The G-21 System

The G-21 multi-processor system at CIT currently serves as the main computer facility of the Computation Center. As such, it includes storage and input/output devices of varied function and capacity. Figure 2 shows the G-21 components most relevant to the display system. The main memory is housed in eight independently accessible memory modules (MM) of 8K, 32-bit words. Each module is electrically coupled to the two central processors (CP) by means of a common "party line" circuit called the "processor bus". All communications between the memory modules and the standard G-21 input/output equipment must utilize the processors and the processor bus. No direct channels to memory are provided. Equipment requiring high channel capacities, such as the display consoles, can communicate directly with either of the two specially adapted modules labeled MM* in Figure 2. A MM* module serves as the regeneration memory for the consoles. No other buffer memory is provided. Thus, the display regeneration memory and standard main memory are indistinguishable by the processors.

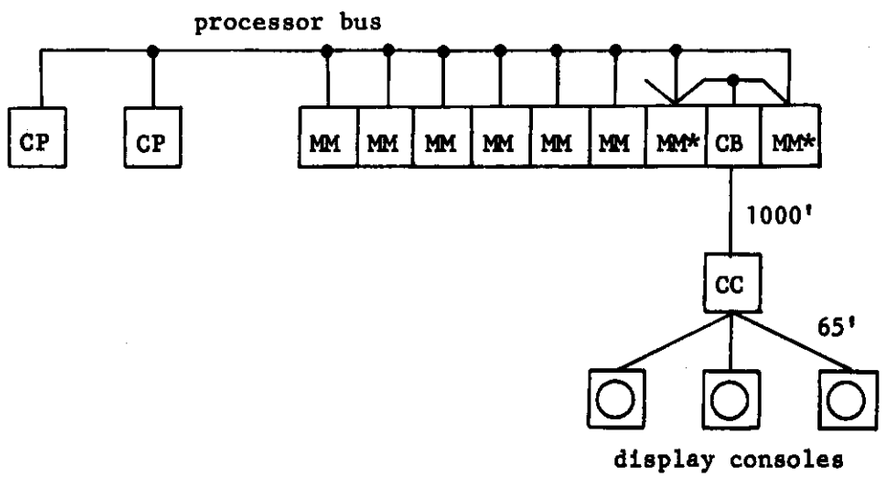


Figure 2

RELATIVE COMPONENTS OF THE G-21 SYSTEM

The Memory

The regeneration modules were adapted to display equipment use by a redesign of the module accessing electronics. They have been given three entry portals, only one of which is the standard processor bus. A detailed description of the G-21 accessing system can be obtained from reference [1]. For the purposes of this paper, it is sufficient to note that the display equipment can access any word in one 8K module without interfering with accesses of the remaining 56K of main memory. Coincident access requests by the display equipment and the processor bus may be assumed to be resolved in favor of the display equipment every other memory cycle. This means that the processors cannot exclude the display equipment from a sequence of memory accesses. Conversely, in the worst

case, the display equipment extends processor access cycle durations from the standard 6 μs. to 12 μs. Normally, processor cycle extensions will be rare since the modules are independently accessible and the MM* module is reserved for display regeneration. Extensions, or "cycle stealing", occur only when the processors are communicating with the display equipment.

In general, display communications are of two kinds. Either the processors are exchanging blocks of information to be displayed, or they are engaged in computational activity dependent upon the displayed information. In the first case, the extended 12 μs. cycle is adequate for all G-21 storage and input/output equipment (viz. 120 KC tapes, 50 KC disc, and 200 KC bulk magnetic card memory). The speed decrease is noticeably large only when large blocks of data are being transferred into the regeneration module from some other module. In the second case, the speed decrease is never noticeably large. Programs, non-displayed data, and scratch work data are not normally stored in the regeneration mode.

The relative independence of the regeneration module stems from this minimization of "stolen cycles". The minimization is dependent upon two features of the system. First, the modules are independently accessible. The display equipment described here will function in any computer system in which the modules are accessible through an independent channel. Second, the display equipment is designed to utilize the independence of regeneration memory. The internal representation of displayed information, and the means by which the console operator interacts with the

displayed information, relieve the processor of the necessity to closely attend the display equipment.

The Display Equipment

The regeneration memory is coupled to the display consoles through the devices labeled CB and CC in Figure 2. Restrictions on processor bus cable lengths require CB and all memory modules to be in physical contact with each other. In order to locate the consoles conveniently, the memory module signals must be reshaped and deskewed for transportation over long cables. The Console Booster (CB) serves the single purpose of interfacing a 1000 foot cable. At the other end of the cable, the Console Controller (CC) channels the information to the appropriate console. The purpose of the CC is to house, in one chassis, all common electronics which can be time-shared by the consoles. Among these are the control sequencers, data registers, character generator, and format generator. The vector generators are not time-shared. Because of bandpass limitations, each console is equipped with its own vector generator. The specifications for the character generator, vector generator, and format generator are given in the first three appendices.

Time-sharing of the CC circuitry is controlled by the regeneration data. In effect, all consoles present and accept all information processed by the CC. However, the regeneration data causes CRT blanking and switch blinding so that each console is responsive to its own information only.

Each console is connected to the CC by means of a 65 foot cable. One CC accommodates four consoles although only three are being installed initially. Each MM can be coupled to one CC so that a theoretical maximum of 32 consoles is possible. Expansion to half that number would probably totally absorb the G-21.

The consoles consist of a collection of independent devices separately cabled to the console main frame. As shown in Figure 3, the collection constitutes a "do-it-yourself console kit" which can be arranged to suit the operator and the task. The detailed operational description of the console components appears, in this paper, after the section describing the internal data representation. The specifications appear in the appendices. The definition of each component appears below.

The Console Components

Figure 3 is a schematic representation of the console components. The console main frame houses the CRT, the CRT image controls, a set of 32 switches called the "state switches", and a pair of 6-bit digital shaft encoders. Each is specified in Appendix 4. The display area of the CRT is approximately 9 1/2" x 9 1/2". Control circuits treat the display area as if it were a 1024 x 1024 coordinate system of display points. The abscissa is referred to as "X" and the ordinate as "Y". The point (0,0) refers to the upper-left corner. The standard character size is 16 x 16 (although other sizes are provided). With one full character height between lines of text, the tube capacity is 32

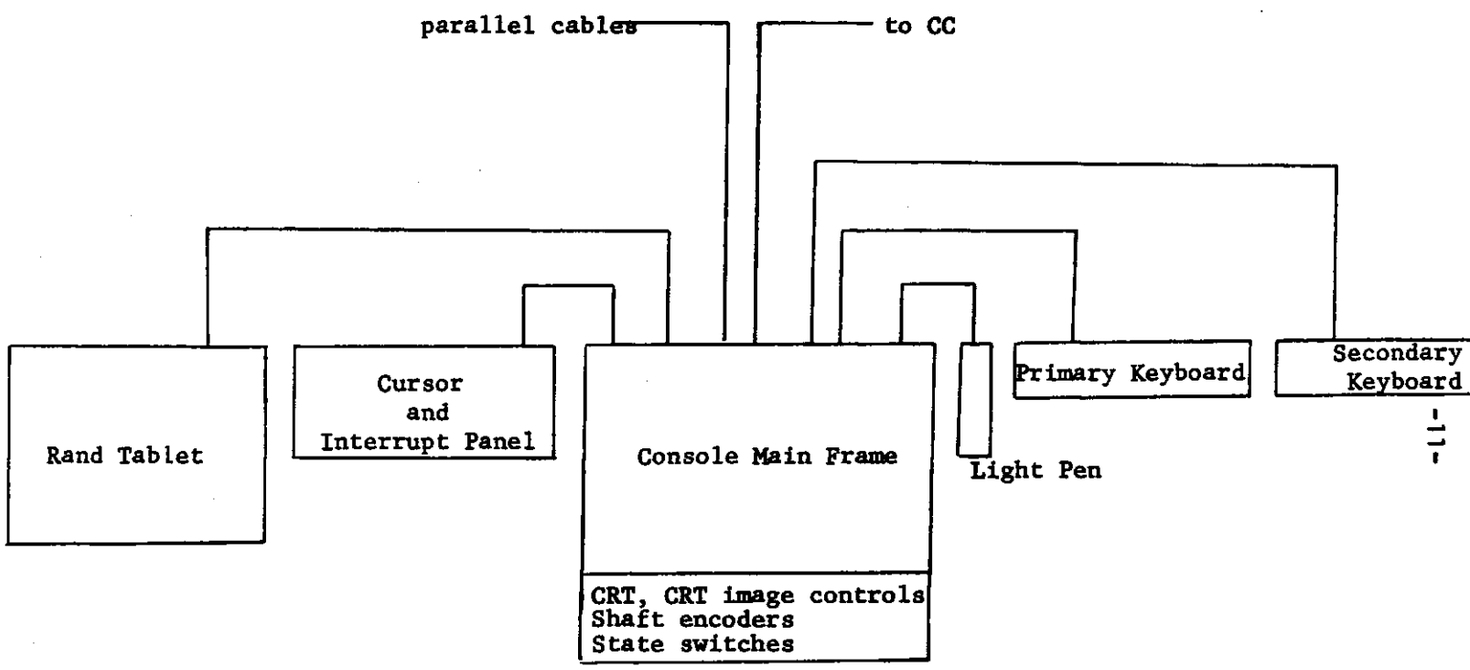


Figure 3

DISPLAY CONSOLE COMPONENTS

lines of 64 characters each. The CRT image controls adjust absolute character dimensions, overall image dimensions, image centering, and spot size. The state switches define console operations modes. They communicate directly with the regeneration memory so that the console state can be both monitored and established by G-21 program. In this sense, the G-21 and the operator are partners in creating an operation mode. Whatever one can do, so can the other. The shaft encoders are context free. They may be read and interpreted by the G-21 program.

The cursor and interrupt panel houses twenty interrupt push-buttons and seven cursor control switches. It is specified in Appendix 5. The interrupt push-buttons can be used by the operator to communicate with his programs on an interrupt basis. They generate a 20-bit code which is loaded by the CC into a reserved cell of the regeneration memory. The setting of any bit causes a single processor interrupt which notifies the G-21 program to read and interpret the 20-bit interrupt code. A second interrupt will not be generated by a depressed interrupt push-button until it is released and depressed again.

The cursor control switches function as a mechanical locator. Four of the seven switches are notated by an up, down, left, or right arrow. When one is depressed, registers in the CC are incremented or decremented according to the direction of the arrow. A special "cursor" symbol appears at the CRT coordinates defined by the contents of the registers. The increment or decrement value is selected by the fifth and sixth

switch. The fifth switch, notated COARSE/FINE selects the values 16 and 1 respectively. Thus, the cursor can be forced to move exact multiples of character dimensions, or it can be enabled to move to any coordinate position on the tube face. All console operations which involve character positioning automatically truncate the coordinate values to avoid cursor pointing ambiguities.

Each cursor position change requires one full cycle through regeneration memory. At marginal flicker, this limits the cursor velocity to 32 steps per second. In order to translate the cursor across the entire tube face, the FINE increment would require up to 33 seconds and the COARSE increment up to 2 seconds. The sixth cursor control switch, SLEW, serves the purpose of skipping the cursor across the tube face at 32 coordinate units per increment. It overrides the COARSE/FINE switch and insures a diagonal traverse time of 1 second at marginal flicker. It is placed so that the operator may retain finger contact with the direction switches while depressing the SLEW switch with his palm.

The seventh switch, MARK, is similar to "pen-down" on the Rand tablet. It is used by the CC to distinguish between an intermediate and final cursor position.

The registers driven by the cursor control switches are shared by the Rand tablet and the light pen. In this sense, the three locating devices are interchangeable. The Rand tablet is specified in Appendix 6. Initially, only one is being prepared for the system. It is housed in a cabinet mounted on casters. The cabinet may be rolled up to any

console and cabled to the main frame.

The two keyboards and the character set are specified in Appendix 7. The total character set consists of 256 distinct symbols, 250 of which are enterable from the keyboards. Except for the quantity of keys and characters, the keyboards are organized much like a standard typewriter. The Secondary Keyboard contains 67 keys including control keys such as up-shift, carriage return, etc. The Primary Keyboard contains an additional key, "underline", which is analogous to up-shift. The non-control keys produce one code during up-shift and another during down-shift. On the Primary Keyboard, a sub-set of 30 keys produces a third code when "underline" is depressed. In order to ease the independent use of the two keyboards, 28 characters appear on both. The keyboard encoding and the character set were selected to produce the greatest number of distinct symbols enterable from currently available keyboards.

Figure 3 shows a line marked "parallel cables". The purpose of these cables is to allow a second set of interrupt push-buttons and a second CRT to be installed in parallel to those of a display console. This second set constitutes a "sub-console" which can be used for experimentation, hard-copy reproduction devices, or image projection and control. The "sub-console" is not yet developed. Only the cable connectors and electronics will be included in the initial system.

The device which integrates the console components into a single operational unit is the regeneration memory. In essence, the regenera-

tion memory is a central switching mechanism which buffers and routes information within a collection of devices including the central processors. It represents only a fuzzy boundary between display equipment and standard G-21 equipment. Boundary fuzziness is also apparent at the operator/system interface. All except protected operations can be initiated by either the operator or the G-21, since all operations are channeled through the regeneration memory.

DATA REPRESENTATION IN THE REGENERATION MEMORY

Memory Organization

Information is stored in one of two regions, in regeneration memory, according to the method of retrieval. The region of memory called the "system" region is allocated to the consoles by a fixed address mapping. The memory addresses of the "display" region hold no fixed relationship to the consoles. By means of a coded set of parentheses, called the "delimit command", information stored in the display region is able to allocate blocks of display cells to various combinations of console components. The CP program is thereby able to organize the display region as is needed. Any console may be assigned any length block up to the module maximum. The block is relocatable since the data within the block is address free. Any combination of display tubes may be assigned to any delimited block; but only one set of input devices may load one block.

Assignments by the delimit command are based upon the information stored in the "system region". Addresses stored in the delimit command

are, in fact, addresses of system cells.

The System Region

Figure 4 shows the allocation of system cells. The first octet is reserved for CC scratch area. It must not be written into by a CP. The next four octets are assigned, in order, to the four consoles attachable to the CC. For ease of electronic addressing, the next four octets are unused. The CP and any new devices are free to write into unused system cells. The last four octets are equivalent to the first four console group octets. They serve as an alternate set of console group octets thereby permitting the system programs to display messages without disturbing the operator's console state.

Figure 5 is a detailed breakdown of the console group referenced by Figure 4. The state word is an aggregate of 32 independent bits corresponding to the 32 momentary-contact state switches. The leading edge of each state switch signal complements an associated flip-flop of a 32-bit buffer register and sets bit 33 of the register. Each register bit drives the indicating lamp of the corresponding state switch. The state of the console is defined by the state of the buffer register.

Whenever the CC encounters the head cell of a text block, the contents of the appropriate state word and the contents of the appropriate buffer register are equated according to the state of bit 33. If bit 33 = 0, then the contents of the state word replaces the contents of the buffer register. If bit 33 = 1, the reverse operation occurs and bit 33

OCTAL ADDRESS	ALLOCATION
0 0 n	Temporary storage used by the CC electronics (not to be used by the CP)
0 1 n	Primary console group No. 1
0 2 n	Primary console group No. 2
0 3 n	Primary console group No. 3
0 4 n	Primary console group No. 4
0 5 n	Unused
0 6 n	Unused
0 7 n	Unused
1 0 n	Unused
1 1 n	Alternate console group No. 1
1 2 n	Alternate console group No. 2
1 3 n	Alternate console group No. 3
1 4 n	Alternate console group No. 4

Legend: n = 0, 1, 2, 3, 4, 5, 6, 7

Figure 4

THE ALLOCATION OF SYSTEM CELLS

is reset. Thus, the G-21 program may load the buffer register, via the state word, during regeneration cycles in which the state word is not being loaded by the state switches.

The interrupt word records display generated interrupt conditions. The display equipment generates three classes of interrupts. Bit 31 is set, and an interrupt is generated whenever a console operation overflows

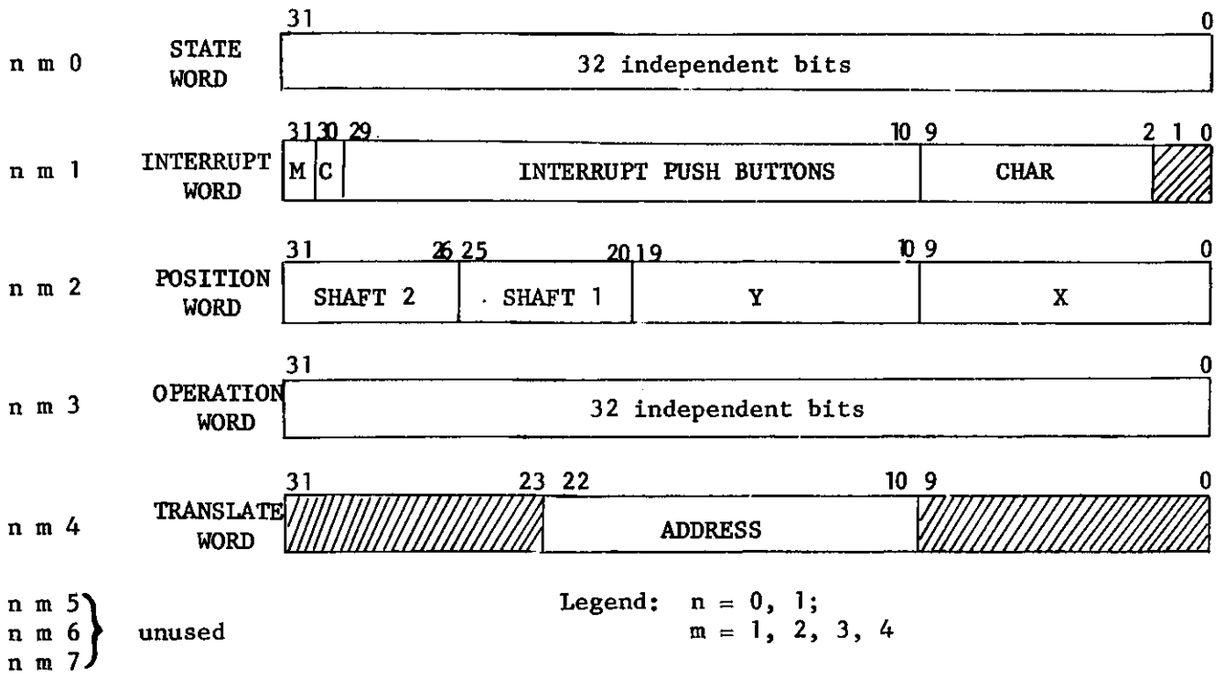


Figure 5

THE CONSOLE GROUP

the allotted block of display memory. Bit 30 is set by a display region command called "compare". The purpose of compare is to generate an interrupt whenever a specified character is entered from the keyboard. The eight bit CHAR field holds the code which caused the most recent compare interrupt. The 20-bit INTERRUPT PUSH-BUTTONS field is a copy of the interrupt push-button state. It is refreshed by an occurrence of the delimit command.

The position word is a copy of the two positioning devices: the shaft encoders and the cursor register. A 6-bit field corresponds to each shaft encoder. A 10-bit Y and a 10-bit X field corresponds to the coordinates of the cursor symbol. The position word is refreshed by the delimit command.

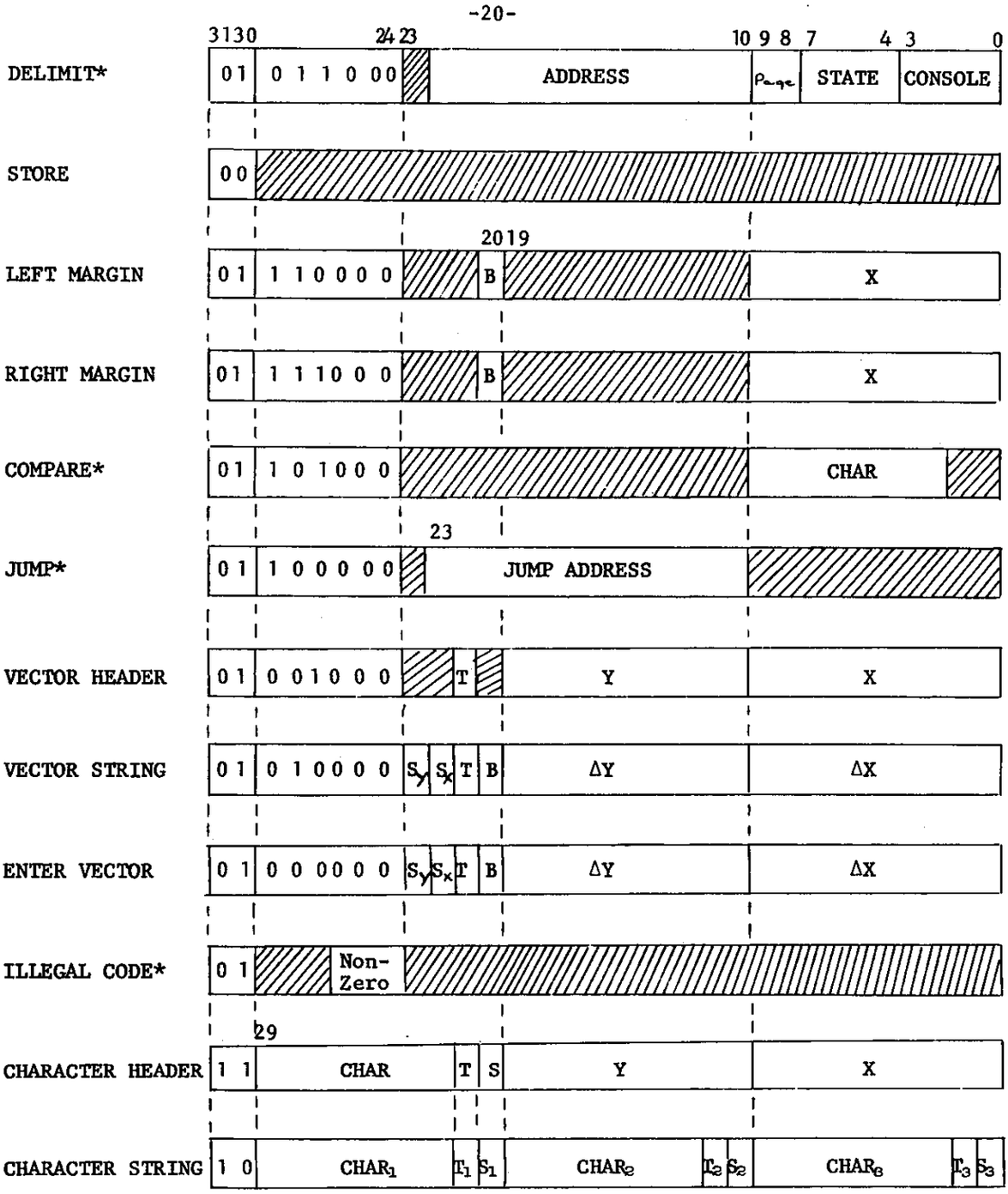
The operation word is reserved for temporary CC storage of the command being executed. The translation word is reserved for the address of the command being executed.

The remaining three console group words are unused. Their purpose is to simplify electronic addressing of the console groups. Any device is free to use them as the need arises.

The Display Region

The region begins at octal location 150 and ends at octal location 17777 (see Figure 1). Each word within the region is interpreted by the CC as one of the twelve opcodes shown in Figure 6. The most significant two bits of each word determine its opcode classification at either STORE, general opcode, CHARACTER HEADER, or CHARACTER STRING. Cross-hatched bit positions are ignored by the control circuitry. The remaining bit positions and the configuration of the state switches determine the details of the opcodes described below. A summary of state switch interpretations is given in Appendix 4.

DELIMIT - The DELIMIT opcode establishes text block boundaries, selects consoles, and prepares the CC for communication. Each text block



* Denotes codes which cannot be generated from the console.

Figure 6

THE CC OPCODES

extends from its DELIMIT opcode to the address specified by the ADDRESS field of its DELIMIT opcode. Every scanned occurrence of DELIMIT causes the ADDRESS field to be stored in regeneration memory location 1 (along with the entire DELIMIT command). When communications with the text block cease to be useful, the CC executes a sequence jump to 1+ (the address specified by location 1). The G-21 program can determine which text block is active at the moment by reading location 1.

The four-bit CONSOLE field selects which of the four consoles are to be active for the forthcoming text block. Any combination of the four is permissible. (Thus, common messages are possible.) Each text block is assigned a page number by means of the two-bit PAGE field. Any combination of pages can be addressed by the PAGE 0, PAGE 1, PAGE 2, and PAGE 3 state switches of the selected console. (The user may alternate between any of four pages, without computer intervention, or he may obtain up to a four-level overlay of pages.) If no console is specified, or if the specified page is not selected by the state switches, the text block is skipped by a CC sequence jump to 1+ ADDRESS. All opcodes within the skipped text block, including DELIMIT, remain unscanned.

The four-bit STATE field specifies the state of the selected console by addressing the system cells as shown in Figure 7. Each encoding specifies one of two input modes and one of the eight console groups shown in Figure 4 and Figure 5. Eight console groups are provided so that each of the four consoles may have a "primary" state and an "alternate" state. The alternate state permits general system messages to be displayed on

CODE	MEANING
7654	
0000	Associate primary console group 4 with the selected console and inhibit
0001	1
0010	2
0011	3
0100	4 and enable
0101	1
0110	2
0111	3
1000	alternate 4 and inhibit
1001	1
1010	2
1011	3
1100	4 and enable
1101	1
1110	2
1111	3

Figure 7

The STATE Field of the DELIMIT Command

all four consoles without disturbing the user's primary state. The two input modes are "enable" and "inhibit". Enable mode permits input data from the selected console to modify data of the text block. Inhibit mode protects the text block from console input by inhibiting the function of the STORE opcode described next.

In addition to defining the forthcoming text block, scanned occurrences of DELIMIT initialize the margin settings to full left and full right. The beam position registers are given values corresponding to the upper left corner, and the cursor position is restored from the position word (Figure 5). Updating of the state word and the state

switch buffer register proceeds as previously described.

As stated in Figure 6, the DELIMIT opcode cannot be generated from the console. It is one of the four opcodes which reserve highest level control for the G-21 program.

STORE - In order to permit expansion of the text block as information is received from the console, the text block is partitioned by the STORE command into an "occupied" and an "unoccupied" region. The occupied region begins at the location of DELIMIT and extends to the location preceding STORE. It contains all text block data which is considered useful. The unoccupied region extends from the location of STORE to ADDRESS. The CC automatically skips to ADDRESS +1 upon scanning an occurrence of STORE. Thus, opcodes located beyond STORE are not scanned by the CC.

Input operations which expand the occupied region do so at the expense of the unoccupied region. In this way, the list-block growth process, described in the next section, can expand the occupied region of a text block without expanding the text block itself. Expansion is accomplished by overlaying the first occurrence of STORE and clearing to zero the contents of the following cell. Thus, the STORE command is translated through the text block by input operations until location ADDRESS +1 must be cleared. Instead, the memory full interrupt is generated and the text block ceases to contain an instance of the STORE command. Subsequent input operations will generate memory full interrupts when information is lost.

LEFT MARGIN - An occurrence of LEFT MARGIN causes the loading of the

contents of the X field into the left margin register in the format generator. Carriage returns cause the replacement of the X display position register by the contents of the left margin register. Whenever B = 1, the CC ignores the LEFT MARGIN command and leaves the contents of the left margin register unchanged. The control character "[L]" can be displayed, at each occurrence of LEFT MARGIN, whenever an input operation requires a reference point for the cursor.

RIGHT MARGIN - An occurrence of RIGHT MARGIN causes the loading of the contents of the X field into the right margin register in the format generator. The CC automatically compares the contents of the X display position register with the contents of the right margin register. Whenever the position of a typed-in character is within 10 character positions of the right margin, the tone is sounded. Whenever a displayed character is within one character position of the right margin, an automatic carriage return is simulated. The B field is similar to that of LEFT MARGIN. The RIGHT MARGIN control character is "[R]".

A margin release character "[M]" is enterable from the keyboard. An occurrence of the release character inhibits both margin opcodes until either a carriage return is encountered, or until X $\geq$ 1023.

COMPARE - If the character specified by the CHAR field agrees with the character generated by the selected console, a character interrupt is generated, and the character is loaded into the CHAR field of the interrupt word (Figure 5). Only one interrupt is generated for each typed character. Interrupts conditioned by characters which cannot be

entered from the keyboard (viz. BLANK, "[L]", etc.) are never generated. The COMPARE opcode cannot be generated from the console.

JUMP - An occurrence of JUMP causes the CC to continue sequencing through memory at location JUMP ADDRESS. The JUMP opcode cannot be generated from the console.

VECTOR HEADER - An occurrence of VECTOR HEADER causes the vertical and horizontal vector position registers of the selected console to be loaded by the contents of the Y and X fields, respectively. The T field is a means of tagging the VECTOR HEADER word and may be interpreted by the G-21 program.

VECTOR STRING - Occurrences of VECTOR STRING cause the vector generator of the selected console to draw a line from the point defined by the position registers to a point displaced by (ΔX, ΔY) on the tube face. The sign of the increments are determined by the value of S_x and S_y respectively. The value "zero" corresponds to a positive sign. The T field is a tag which can be interpreted by the G-21 program. The B field blanks the cathode during vector drawing so that the drawn line segment is invisible.

The contents of the position registers are incremented by the ΔX and ΔY values. The incrementation is digital so that no analog error accumulates.

ENTER VECTOR - The ENTER VECTOR code is reserved for CC use during vector input operations which span several cycles through the text block.

ILLEGAL CODE - Unless bits 24, 25, and 26 are zero, opcodes with

bit 30 = 1 and bit 31 = 0 are interpreted as illegal codes by the CC and ignored. Illegal codes may be distributed throughout regeneration memory by the G-21 program. Each word provides 27 bits of context free storage which can serve as names, tags, etc.

CHARACTER HEADER - An occurrence of CHARACTER HEADER loads the vertical and horizontal position registers of the character generator with the contents of the Y and X fields, respectively. The character specified by the CHAR field is then displayed at coordinates (X, Y). The T field is a tag bit which can be interpreted by the G-21 program. The S field specifies the size of CHAR. If S = 0, CHAR is scaled 64 character to the line. If S = 1, CHAR is scaled 32 characters to the line. After CHAR is displayed, the contents of the X position register are incremented by 16 if S = 0, and by 32 if S = 1.

CHARACTER STRING - An occurrence of CHARACTER STRING causes the three characters $CHAR_1$, $CHAR_2$, and $CHAR_3$ to be loaded in the given sequence. Generation and display is as described above except that the position registers are not initially loaded.

When the right margin is encountered during CHARACTER STRING, a carriage return is simulated. The contents of the Y position register are automatically incremented by 32 regardless of the given S value. Thus, normal line spacing provides 32 lines to the displayed page. With S = 0, line spacing is adequate to provide one unused line between every used line. With S = 1, characters on consecutive lines may touch.

DATA CONTROL BY THE CONSOLE OPERATOR

The State Switches

The state switches establish the operating state of the console. They are the means by which the console operator controls input processes and display options. In effect, the state switches select which opcode interpretation is valid and which opcode formats are to be associated with incoming data. Once entered, the incoming data and opcodes provide a context in which operating states are re-interpreted. Thus, the internal representation permits complex input processes to be controlled by the CC rather than the human operator. For example, the initial input of a new figure and the modification of an existing figure are both controlled by the INSERT VECTOR switch. The difference between the two processes is established by the internal representation and need not concern the operator.

Figure 8 shows the layout of the state switches. Further details are given in Appendix 4. The state switches are physically located on the main frame (see Figure 3). There are indicating push-button switches arranged in a four by eight array. The rectangular lenses are engraved with state names which are visible in both indicating and non-indicating states.

As previously described, all state switches have momentary contacts which drive a state switch register which, in turn, drives the indicating lights. The operator need not be aware of this underlying hardware. For most switches, depressing once turns the switch "on". A second push turns

PAGE 3	PAGE 2	PAGE 1	PAGE 0
DISPLAY CONTROL SYMBOLS	DISPLAY and SPACE	SUPPRESS OVERLAY	SINGLE SPACE
BLINK	INTENSIFY	X ONLY	Y ONLY
ENTER LEFT MARGIN	ENTER RIGHT MARGIN	ENTER CHARACTER	ENTER VECTOR
DELETE	REPLACE	FLAG	UNFLAG
INPUT WITH TAG	INPUT DOUBLE SIZE	CLEAR	
3	2	1	0
7	6	5	4

Figure 8

THE STATE SWITCHES

the switch "off". Most switches are independent of each other. Some, such as FLAG and UNFLAG, are not only dependent but also mutually contradicting. The only logic apparent to the operator is that which corrects switch contradictions. For example, switching FLAG "on" automatically switches UNFLAG "off".

Output

The upper-most three rows of state switches select display modes. Their use in no way disturbs information resident in regeneration memory.

PAGE - The top row selects, for display, any combination of the four pages available to each console. In effect, each page switch enables recognition of the appropriate delimit command. If no page switch is depressed, all text blocks addressed to the console will be skipped by the CC and no display will appear.

DISPLAY CONTROL SYMBOLS - The control symbols such as back-space, carriage return, etc. (listed in Appendix 7) do not normally display. They are recognized and executed by the CC with display inhibited. In order to be modified or deleted, any symbol (including a control symbol) must be selected by the cursor, light-pen, or tablet. Therefore, DISPLAY CONTROL SYMBOL disqualifies the display inhibit circuit thereby making all control symbols visible.

The unintentional selection of invisible control symbols is automatically avoided. The CC recognizes selection of control symbols only if they are made visible by the state switches.

DISPLAY AND SPACE - When control symbols are made visible, combinations such as "front-space, back-space, character" will appear as overlays. The DISPLAY AND SPACE switch is identical to DISPLAY CONTROL SYMBOLS with one exception. All characters, including control symbols, cause one front-space during execution. The display is distorted but no overlays are possible.

SUPPRESS OVERLAY - The DISPLAY AND SPACE switch eases the overlay problem by transforming all overlays into strings. During the construction of overlays, the string transformation can distort the result beyond recognition. Depressing SUPPRESS OVERLAY causes the first occurrence (in memory) of a symbol at the selected character position to become invisible. A second depressing causes the first occurrence to reappear and the next occurrence to disappear, etc. The fifth depressing turns off the SUPPRESS OVERLAY switch indicating light and terminates the operation. The sixth depressing is equivalent to the first.

When the desired overlay appears (i.e., the undesired character disappears) the undesired character can be removed by standard editing operations such as DELETE. The selecting device (cursor, etc.) is interpreted as selecting only the invisible character. Editing operations will not affect the visible characters of the overlay.

SINGLE SPACE - The normal display format provides 64 characters to the line and 32 lines to the displayed page. Normally, one line height is placed between each displayed line to yield a readable double-spaced display. The SINGLE SPACE switch alters the normal format to permit 64

lines to the displayed page. The alteration does not affect the character size. In effect, it changes the Y increment value generated by carriage returns. The resulting display can become intolerably unreadable by such tight packing. It is provided as a means for filling the display when clarity is unnecessary.

BLINK - The tag bit mentioned throughout the preceding sections is program interpreted. However, it also serves two display control functions: BLINK and INTENSIFY. When evoked, BLINK causes all visible tagged characters or vectors to blink. By use of the alternate console group, tagged messages from the computer can be made to blink independent of the displayed text.

INTENSIFY - When evoked, INTENSIFY causes all visible tagged characters and vectors to appear intensified. By using INTENSIFY, bold-face type or depth perception in figures can be achieved.

X ONLY - In all editing states in which the selector (cursor, etc.) must point to a character or to a vector, the symbol located at the selector position automatically blinks for verification. Evoking X ONLY causes all symbols in the same column as the selected symbol to blink so that selector positioning is eased. Verification is correspondingly less accurate.

Y ONLY - When evoked, Y ONLY causes all symbols in the same row as the selected symbol to blink. When both X ONLY and Y ONLY are evoked, a cross-hair of blinking symbols uniquely identifies the selected symbol while easing the repositioning of the selector at a distant symbol.

Input

The lower five rows of state switches select input modes. Unless an input mode is established, display information in regeneration memory cannot be altered or appended. The state switches control input to regeneration memory from the keyboards and the "selectors". A selector is an electro-mechanical locator device such as the cursor, the light-pen, or the tablet.

The interrupt buttons do not have the direct capability of changing the display. Indirectly, they may summon processor programs which effect display changes. The interrupt buttons are always enabled and may use whatever input mode has been selected. However, the lower-most two rows of state switches assist in the definition of interrupt-summoned processor tasks. For that reason, they are considered input state switches.

Some input modes require selector positioning at a currently displayed symbol. If such is the case, all control symbols are automatically displayed and the head cell character of every character string is displayed intensified.

ENTER LEFT MARGIN - In order to establish a new left margin, a selector is located at the desired margin position and the ENTER LEFT MARGIN switch is depressed. The left margin opcode is automatically entered as the next text block word with the selector coordinates as the location parameters, X(see Figure 6).

In addition, depressing ENTER LEFT MARGIN automatically cancels all other conflicting input modes and causes all left margin control symbols

to become visible.

ENTER RIGHT MARGIN - The process for entering right margins is identical to that for entering left margins except that it is evoked by the ENTER RIGHT MARGIN switch.

ENTER CHARACTER - The single input mode for character strings is evoked by the ENTER CHARACTERS switch. After the switch is depressed, a selector can be positioned at any location in the display area in order to define the display location of the incoming character. A "caret" symbol is automatically generated at the selector position for position verification. If the selector position is occupied by a character, the character automatically blinks for more positive verification.

The input character is entered from the keyboard and the cursor is automatically advanced one character position in preparation for the next input. At the end of the line, the cursor automatically advances to the next line as if a carriage return had been generated. When the bottommost line is filled, the cursor disappears, and the input operation terminates. In this way, the entire display area may be loaded as a single string of characters. Previously entered characters may be overlayed or translated by the input process, depending upon the input mode.

Three input modes are obtainable by means of the single ENTER CHARACTER switch. The choice of which mode is implicit in the operator's actions. Consequently, the internal representation of character strings was adapted to permit automatic mode selection.

The first mode corresponds to the initial entry of a character which

bears no obvious relationship to an existing character string. Initial entry is obtained by randomly positioning the selector at any unoccupied character position which is preceded by an unoccupied character position, and striking a key. The CC recognizes the initial entry mode by cycling through the text block, as far as the STORE command, without encountering a character at the position preceding the selector position. The input character appears at the selected position, and the caret progresses to the next position. Internally, the character header code is loaded into memory at the location of the first empty store command (see Figure 6). The following location is cleared to zero to provide a new store command. The selector position is entered as the (Y, X) location and the input character is entered as the first character of a string. The input process then progresses to the string mode.

The string mode corresponds to the entry of appendices to existing character strings. Whenever a character is encountered which is displayed at the position preceding the caret position, the store command is replaced by the character string command, the input character is inserted into the first character position, and the next location is cleared to provide a new store command. The remaining character positions retain the zeros of the original store command. (The all-zeros character is reserved as the non-displaying, non-spacing "blank" character.) As in the initial entry mode, the caret automatically advances to the next character position. Subsequent character inputs will occur in string mode unless the caret is repositioned. Each input character will be placed in the character

string position occupied by the blank character, or a store command will be encountered and translated as previously described. The packing of string characters is thereby established as three per word. Repositioning of the caret will destroy the conditions necessary for the string mode. Depending upon the new caret position, one of the three modes will be defined.

The third mode is internally equivalent to the string mode although appearing different to the operator. If the selector is positioned at a character which is followed by a character, input of a third character will not cause an overlay. Instead, the input character will appear at the selector position and all characters of the modified character string will translate one character to the right. Figure 9 illustrates the string input mode by example. The " * " character was omitted from the original text and later inserted between the " B " and " (" characters. The character at the caret position (" (" in Figure 9) blinks for verification. After input the caret automatically advances to the next character position so that the next character inserted will appear to the right of the previously inserted characters.

Internally, the string input mode is obtained from the block-list representation of character strings. In effect, the block-list behaves like a stack which can be "pushed" at any entry. Characters inserted by ENTER CHARACTER appear in what amounts to "standard form". Once entered into the block-list, character strings input from the keyboards are indistinguishable from those input from the processor. No "patches" are

ORIGINAL DISPLAY A ← B$_{\wedge}$(C + D);

*←INPUT CHARACTER

FINAL DISPLAY A ← B *$_{\wedge}$(C + D);

Figure 9

THE STRING INPUT MODE

necessary.

The stack behavior of the block-list was obtained from the standard memory functions of read and write at negligible cost. All entries are contained in contiguous memory cells of the list block so that character inputs internal to the block increase block length. The input character then appears at the physical memory location which corresponds to its position in the displayed string. Each text block word which follows the input word is displaced by one location after input. The displacement is achieved by the CC during normal cycling through regeneration memory. Displacement begins at the word containing the first displaced character and continues until the store command is encountered.

The displacement process is obtained from standard read and write memory functions by writing the previously read word after every access. The need for shifting characters is eliminated by inserting blank

characters as word fillers. Both displacement and character input then occur on a word basis rather than character by character. In the event that an input character replaces a blank character in the block-list, no displacement occurs. Thus, a string of n inserted characters requires n mod 3 +[n/3] words of regeneration memory.

An example of the displacement process is shown in Figure 10. Originally, the caret is positioned at character "D" which automatically blinks for verification. The input character " α " replaces the resident character "D" and displacement begins. Two blank characters and the "D" form the previous word when word n+3 is read, etc. When the store command is encountered, the displacement process terminates. Input of the character β replaces a blank character so no displacement process occurs.

The initial entry mode differs from the string mode by the formatting of the header opcode and the placement of the header opcode at the store command location. The two string modes are equivalent although displacement is evident only in the mode just described. The previously described mode is the degenerate case where displacement begins and ends at the store command location.

An ENTER CHARACTER operation which eliminates the store command by attempting to relocate it at the next delimit command generates a memory full interrupt without loss of information. Each subsequent input operation, without intervening memory reorganization, will generate a memory full interrupt and <u>will lose</u> information. A standard system message can inform the operator of his predicament.

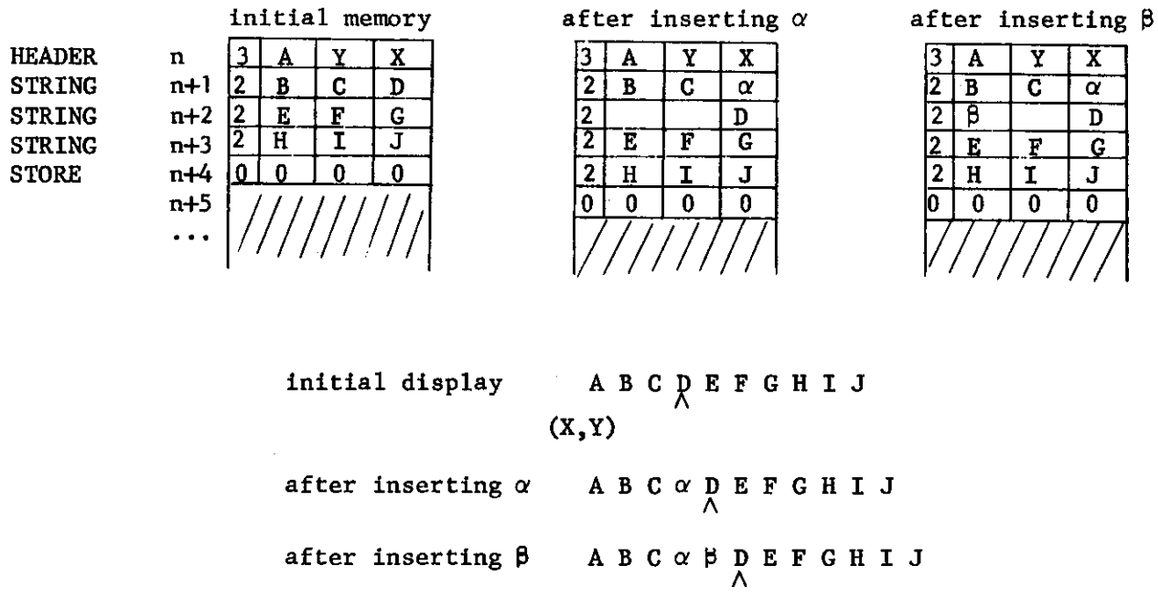


Figure 10

THE CHARACTER DISPLACEMENT PROCESS

ENTER VECTOR - A similar set of input modes is available for vector entry. When ENTER VECTOR is evoked, a figure can be initially entered by positioning a selector and depressing the mark bar on the cursor control panel. The mark bar causes a vector header to be stored in memory with the XY values of the selector position. As the selector is moved from the initial position, a vector string is stored in memory with the ΔX, ΔY values obtained from the selector position. A vector automatically extends from the initial position to the cursor position. The vector string is constantly updated by the selector position so that the dis-

played vector automatically extends to the selector position as the selector is moved about the display area. A second depressing of the mark bar permanently affixes the vector end-point at the selector position. A second vector then extends from the fixed end-point to the selector as the selector is moved about the display area. Subsequent mark bar depressing and selector repositioning allows the figure to be completed. Figure drawing is terminated by "turning off" the ENTER VECTOR switch.

Figures are appended by vector string operations which are equivalent to character string insertions. The example shown in Figure 11 converts a square into a pentagon. As is evident in Figure 11, the displacement process affects all words between the changed block-list and the store command. However, entries of the changed block-list undergo no displacement. Were the figure to be modified by inserting a new vector between vectors 1 and 2, the entire figure would be distorted as shown in Figure 11. Instead, the vector "5" is appended to the figure and deleted later. The deletion process is explained in the description of the DELETE switch. The appending process commences by selecting the last vector string entry. When ENTER VECTOR is evoked, the end-point of the last entry is intensified for recognition. (As with characters, a selected vector blinks for verification.) The figure is then appended by continuation of the "initial entry" process just described.

DELETE - Undesired characters or vectors may be removed from the display by means of the DELETE switch. A symbol is deleted by evoking DELETE and positioning the caret with one of the selectors. The selected symbol

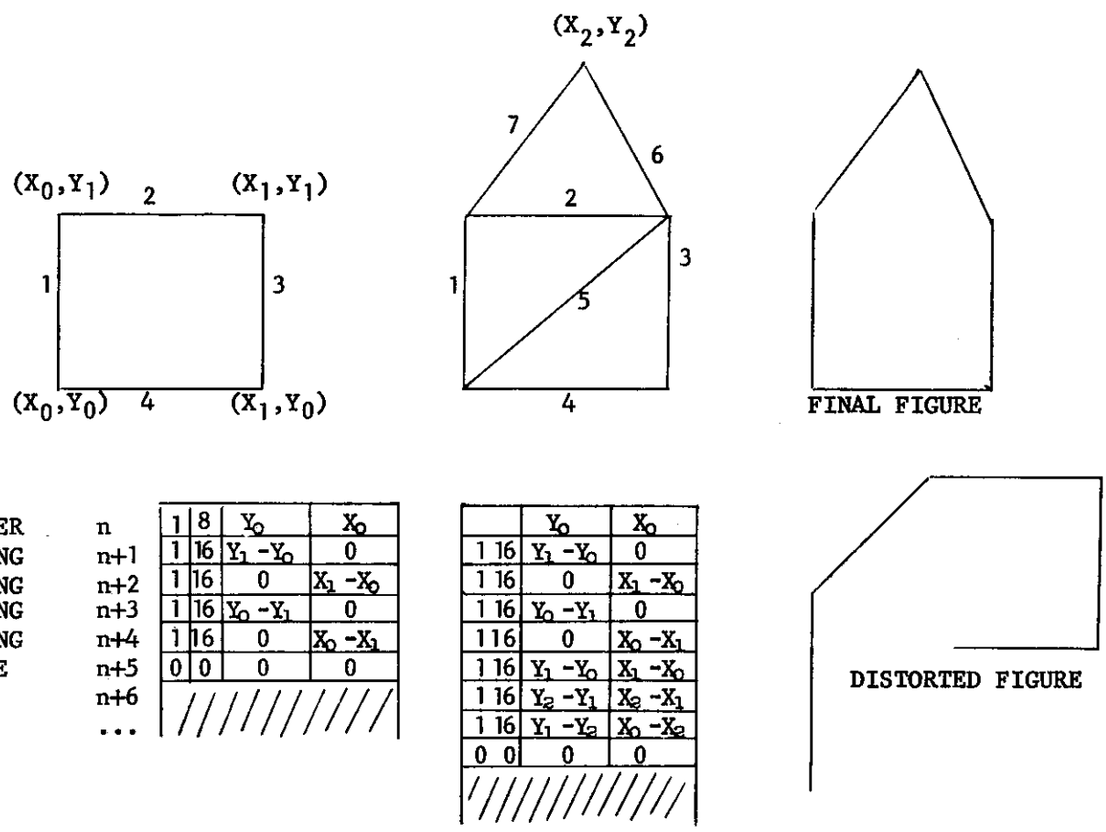


Figure 11

THE VECTOR DISPLACEMENT PROCESS

blinks for verification. Deletion of the symbol occurs when the mark bar is depressed. All control characters are visible, and end-points of all vectors are intensified during DELETE. A vector is selected by selecting its intensified end-point.

The deletion process is interpreted by the CC according to the type

of symbol which is blinking.

Margins are replaced by one of the "no operation" commands so that they are ignored by the CC.

The deletion of a character replaces the blinking character with the blank character without advancing the caret. Character deletion is the inverse of character insertion with displacement, since the blank character does not generate a front-space operation. If the resulting shift of displayed characters is undesired, DELETE should not be used. Instead, the character should be replaced by the front-space character by evoking REPLACE.

The deletion of a vector leaves the vector increments (ΔX and ΔY) intact. Depressing the mark bar inserts a blanking bit (shown as B in Figure 6) which inhibits display of the vector but permits incrementation by ΔX and ΔY. Blanking, as opposed to actual deletion, is of obvious necessity if the figure is to maintain shape during DELETE. The undesired vectors 2 and 5 in Figure 11 are made invisible by DELETE. Invisible vectors may be returned to the visible state by the EXCHANGE operation.

REPLACE - Information in regeneration memory can be replaced by information input from the console without format changes. However, depending upon the selected symbol, two variations of display changes may occur. In all cases, the symbol to be changed is selected in the standard fashion and blinks for verification.

Replacement of character string entries is accomplished by striking

the desired key. The new character appears in both the display and the block-list at the position occupied by the blinking character. Vector string entries may be replaced by moving the cursor to the desired position. The blinking vector will be continually redrawn to terminate at the selector position. Figure distortions will accompany REPLACE operations on vector strings.

The REPLACE operation bears an entirely different significance when header cells are selected. The selection of the initial point of a figure causes the entire figure to translate about the display area as the selector is moved. REPLACE has the significance "RE-PLACE". As with all other vector input operations, once a vector string or vector header word is selected, the CC replaces the location values of the selected word with those of the selector position. Translation of figures is thereby easily achieved.

Similarly, an entire character string can be replaced at any location by "dragging" the initial symbol across the face with the selector. Margins may be reset by dragging the displayed margin symbol to the desired margin location.

In either the vector header or character header case, it is possible for displayed information to be translated off of the tube face. For this reason, the hardware registers corresponding to X and Y positions have twelve bits rather than ten. The additional bits permit strings to translate off of the tube face by exactly one tube dimension in any direction. Translations beyond one tube dimension are impossible from the console

because the selector must always point to the symbol contained in the header cell.

FLAG - If FLAG is evoked, the flag bit of the blinking symbol is set to "one" whenever the mark bar is depressed. If the blinking symbol is a character, a caret automatically advances one space so that an entire string may be conveniently flagged.

UNFLAG - The tag bit is set to "zero" by UNFLAG, in a similar manner to FLAG.

INPUT WITH TAG - When evoked, INPUT WITH TAG causes all input symbols to be stored with the tag value "one".

INPUT DOUBLE SIZE - When evoked, INPUT DOUBLE SIZE causes all input characters to be stored with the size bit value "one". Thus, all input characters appear double size. To change the size bit of a displayed character, REPLACE must be evoked and a new character must be input with the INPUT DOUBLE SIZE switch "off" or "on" as desired.

CLEAR - When depressed, CLEAR remains "on" until turned "off" by the operator. No input is possible while CLEAR is "on". The purpose of CLEAR is to erase the entire memory contents currently displayed. Internally, the store command is loaded into every location betweeen the delimit commands of the displayed text blocks and the following delimit commands.

ALPHABET SWITCHES - The remaining switches have no CC significance. They are used as modifiers for the interrupt switches so that 2^9 interrupt alphabets are possible. As new operations are added to the display

system, the ALPHABET SWITCHES may be converted to standard control switches. They serve as a resevoir of spare state switches.

REFERENCES

[1] Quatse, Jesse T., Design of the G-21 Multi-Processor System, Center for the Study of Information Processing, Carnegie Institute of Technology, February 26, 1965.

[2] Minsky, Marvin, Remarks on Visual Display and Console Systems, Part I, Technical Memorandum No. 1, Project MAC, Massachusetts Institute of Technology, June 14, 1963.

[3] Integrated Memory Display System, TP1833, A Technical Proposal prepared for Carnegie Institute of Technology, Philco Corporation, Willow Grove, Pennsylvania, September 28, 1964.

NOTE

From time to time, the appendices will be modified by correction, addition, and deletion. The modifications will maintain this document as an accurate definition of the described visual display system. Development of the system is not expected to desist in the foreseeable future.

If you wish to receive copies of the modifications, please fill out this page and send it to:

The Engineering Staff
Computation Center
Carnegie Institute of Technology
Pittsburgh, Pennsylvania 15213

NAME _____

ADDRESS _____

_____

_____

_____

NUMBER OF COPIES _____

APPENDIX 1 - THE CHARACTER GENERATOR

The character generator is manufactured by Philco for their READ display system. It is a transistorized control circuit, with a 256 address, read-only, resistor memory, which converts 8-bit addresses into character drawing control sequences. The control circuits construct characters on the display tube as a sequence of at most 24 connected straight line segments. Consequently, it is referred to as a "stroke generator". The resistor memory specifies directions, magnitudes, and blanking for each character. Although it is a read-only memory, the resistors may be changed in the field as the character set is modified or extended.

* Denotes invisible stroke

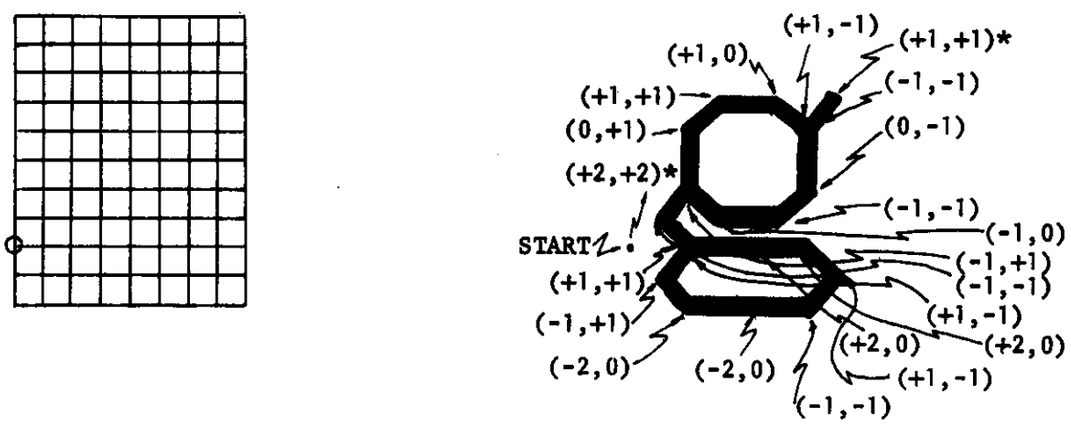


Figure A1-1 The Minor Position Grid for the Character Generator

The control circuits drive a secondary yoke mounted on the CRT. The primary yoke positions the CRT beam at the initial point (the "major position") circled in Figure A1-1. Maximum and minimum excursions from the major position are shown by the grid in Figure A1-1. By means of the secondary yoke, the character generator can produce strokes whose vertical and horizontal ranges are independently 0, 1, 2, or 3 grid lines. Cathode blanking is used to provide invisible links for visibly disconnected segments. The sample character shown in Figure A1-1 requires 21 strokes, two of which are invisible. The major position is used as the base line so that characters, such as "g", may extend two grid lines beneath the base line. Figure A1-2 shows typical character generator "programs" and codes for 256 characters.

Character drawing is an asynchronous operation. The character drawing time depends upon three independent events: major positioning, clock synchronization, and stroke generation. Major positioning and clock synchronization average 2.3 μsec. Each stroke requires 0.3 μsec. The lower case Latin, upper case Latin, digits, punctuation, and parentheses shown in Figure A1-2 (a total of 86 characters) average 12.3 strokes per character. The average character drawing time, for the 86 characters, is 5.98 μsec.

Either of two sizes may be selected by the "size bit" of the character code. The small size (scaled 64 lines to the page) are approximately 1/8" x 1/12". The large size (scaled 64 lines to the page) are approximately 1/4" x 1/6". Absolute sizes may be adjusted by a control knob on the console.

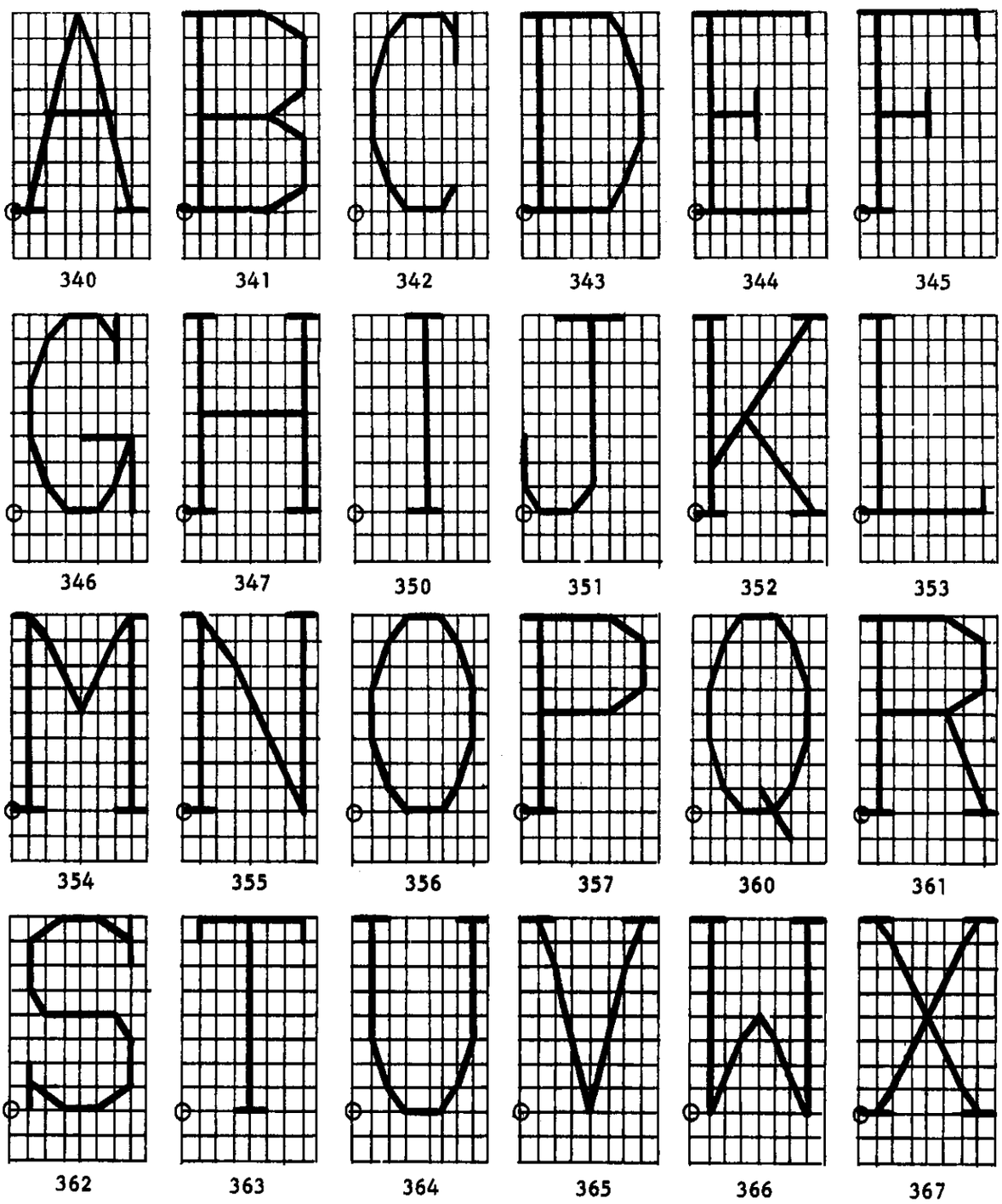


Figure A1-2 Character Generator Programs

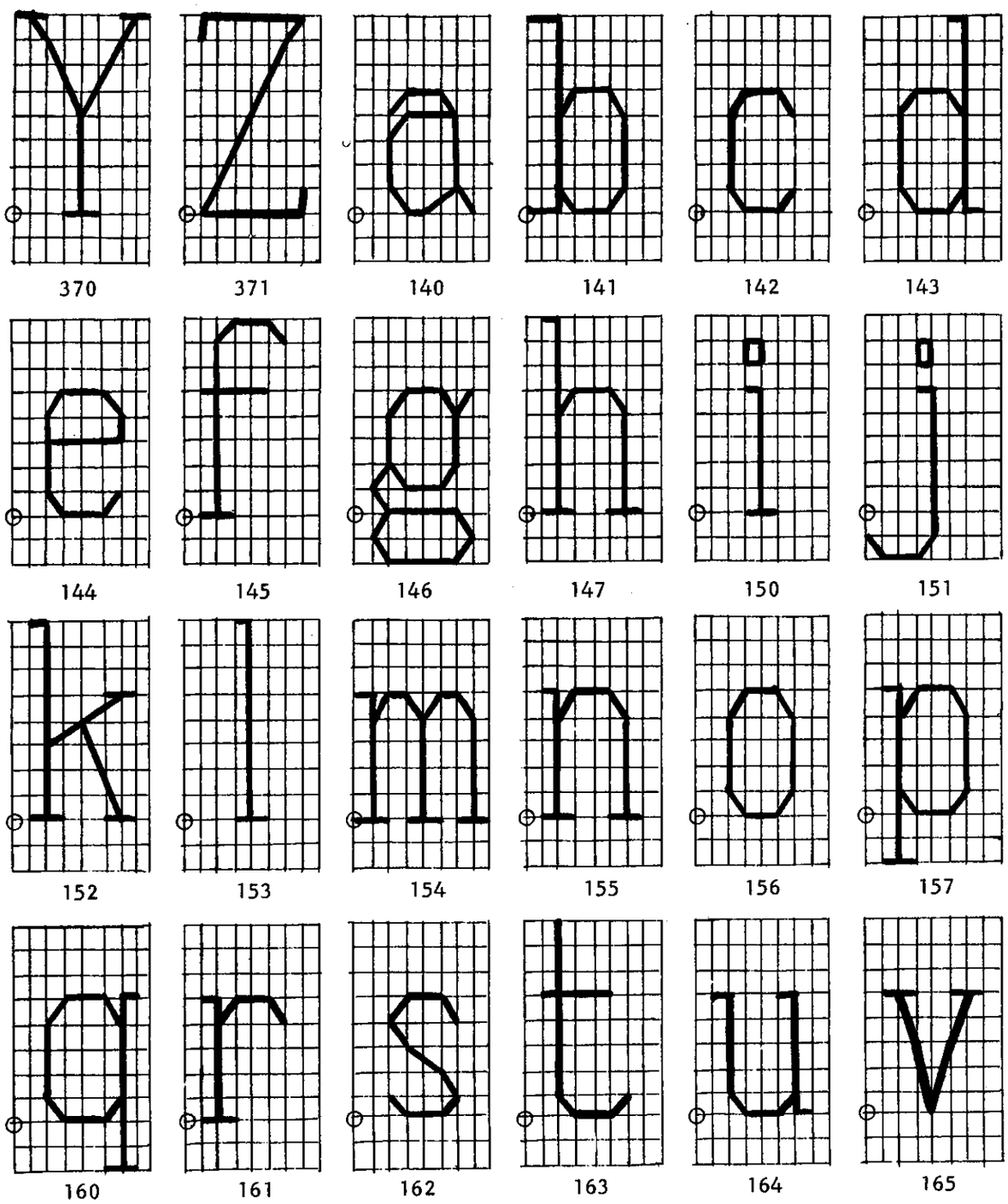


Figure A1-2 Character Generator Programs (cont'd)

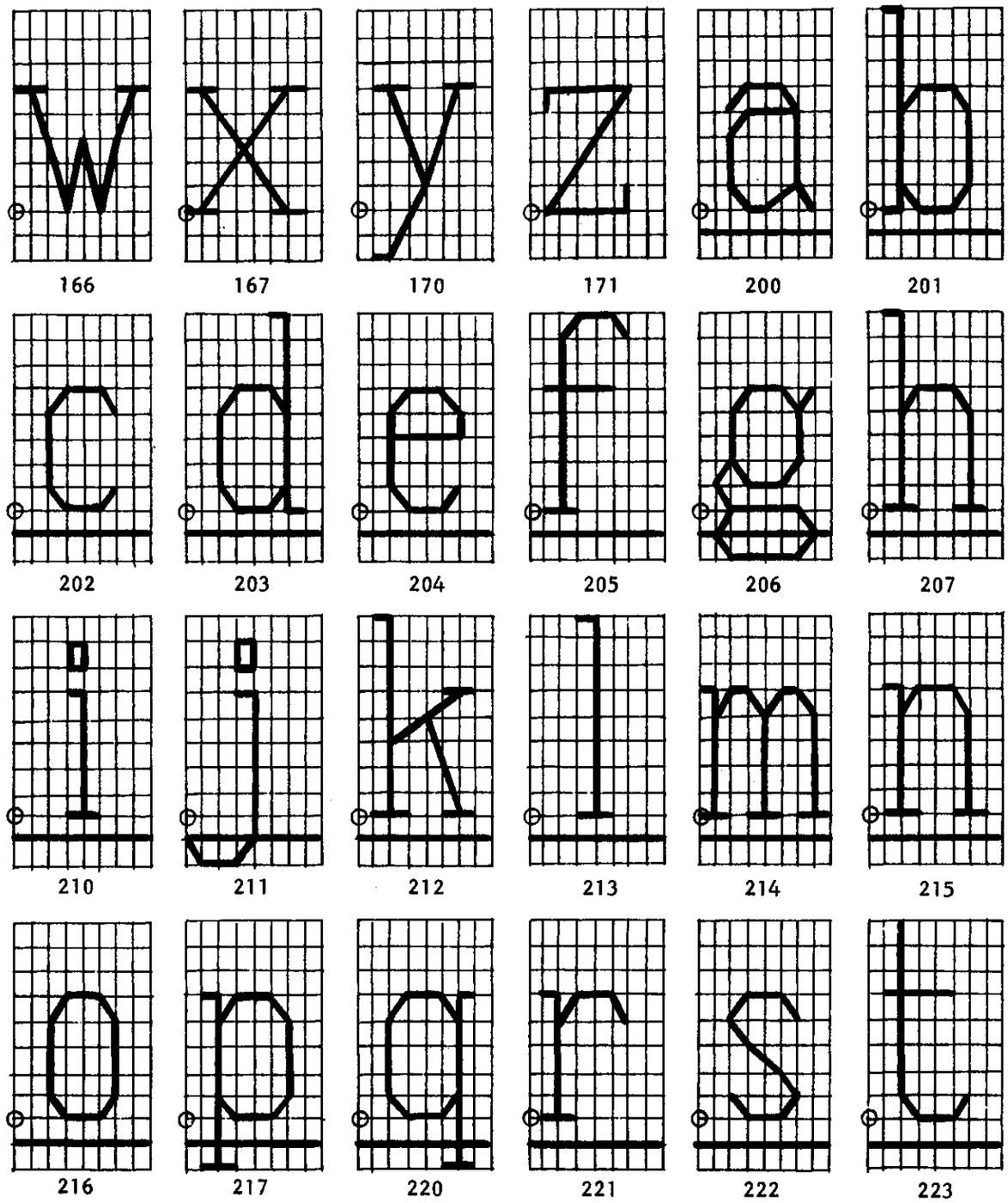


Figure A1-2 Character Generator Programs (cont'd)

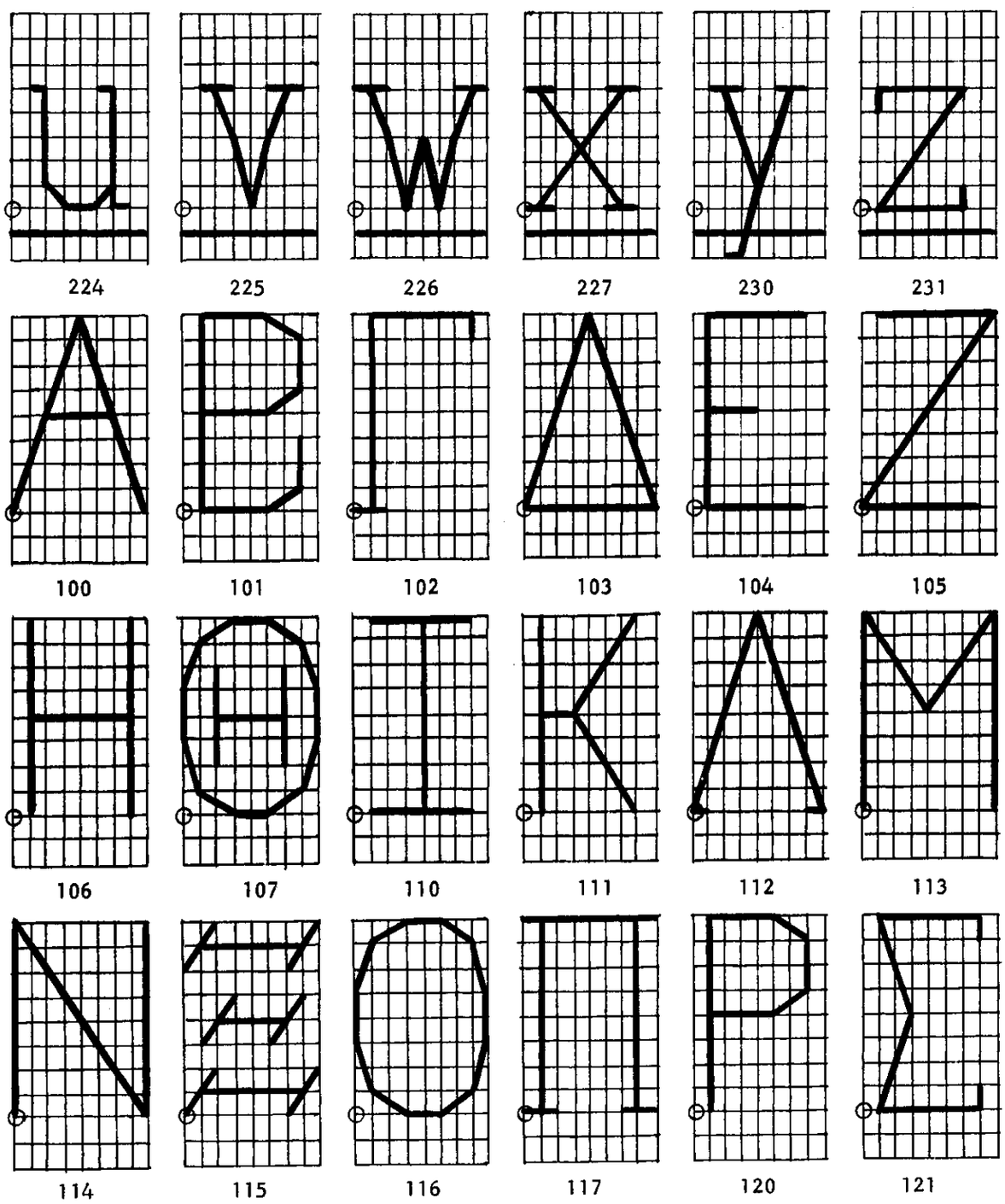
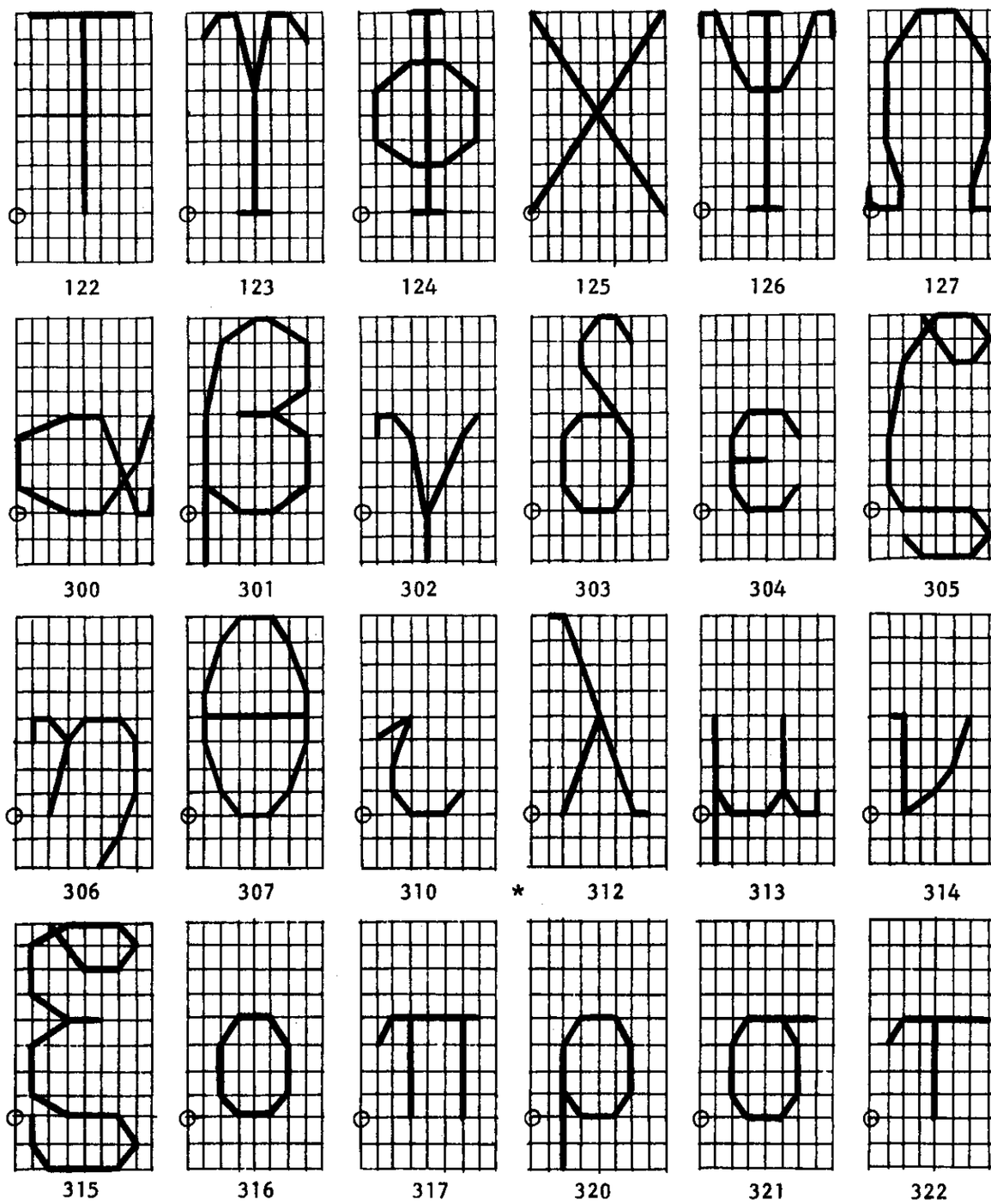


Figure A1-2 Character Generator Programs (cont'd)



* Small case kappa shown on Page A1-13

Figure A1-2 Character Generator Programs (cont'd)

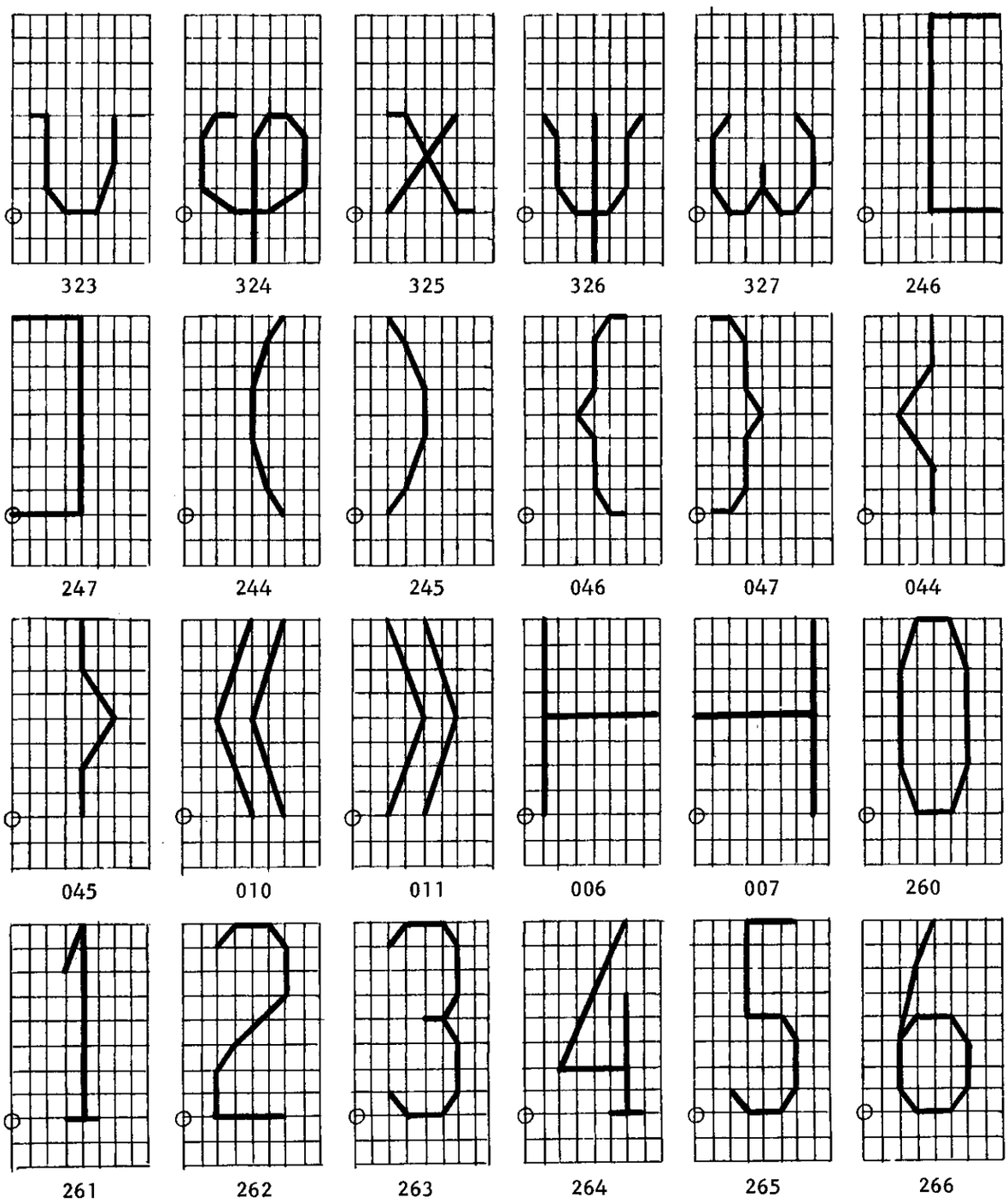


Figure A1-2 Character Generator Programs (cont'd)

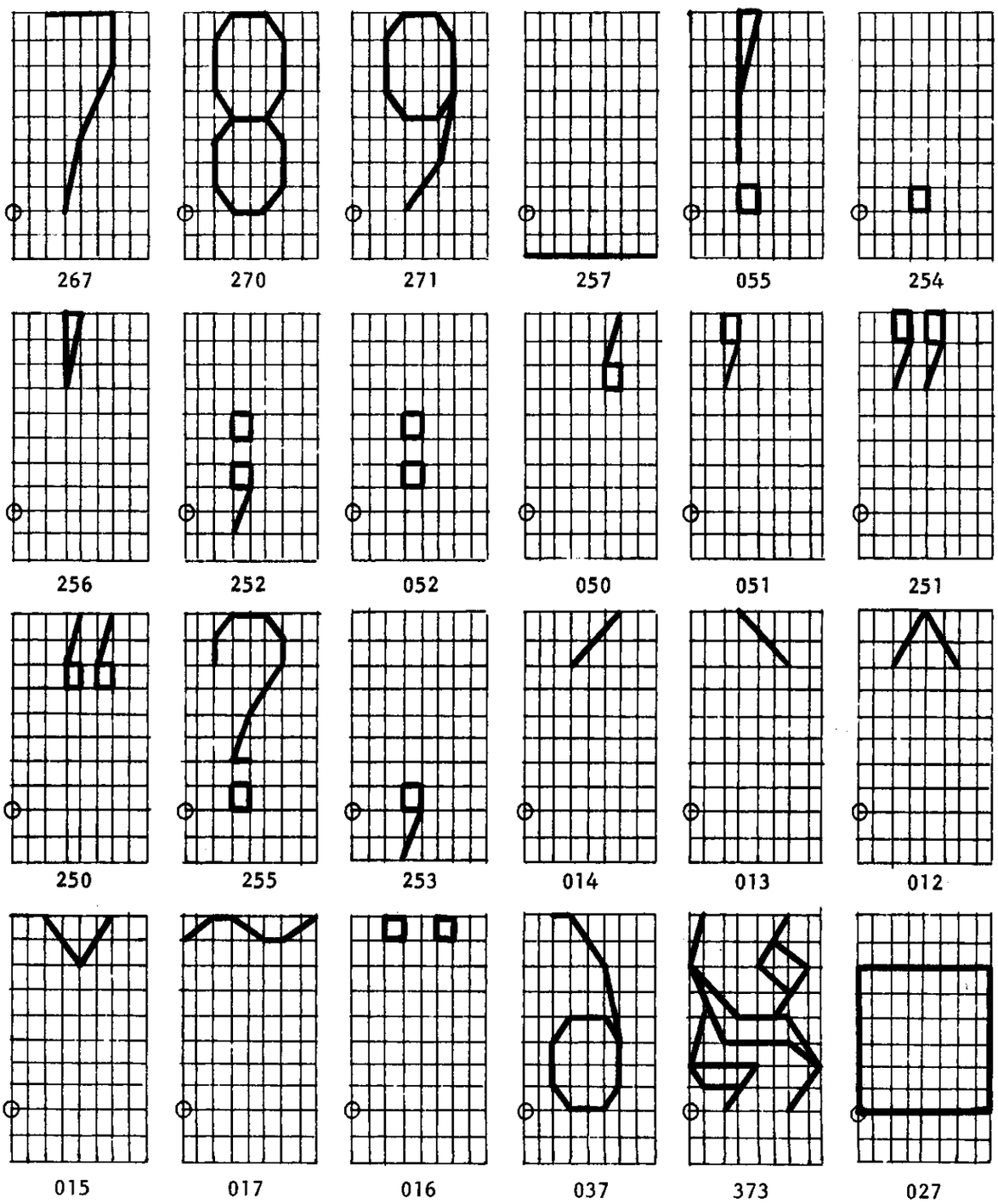


Figure A1-2 Character Generator Programs (cont'd)

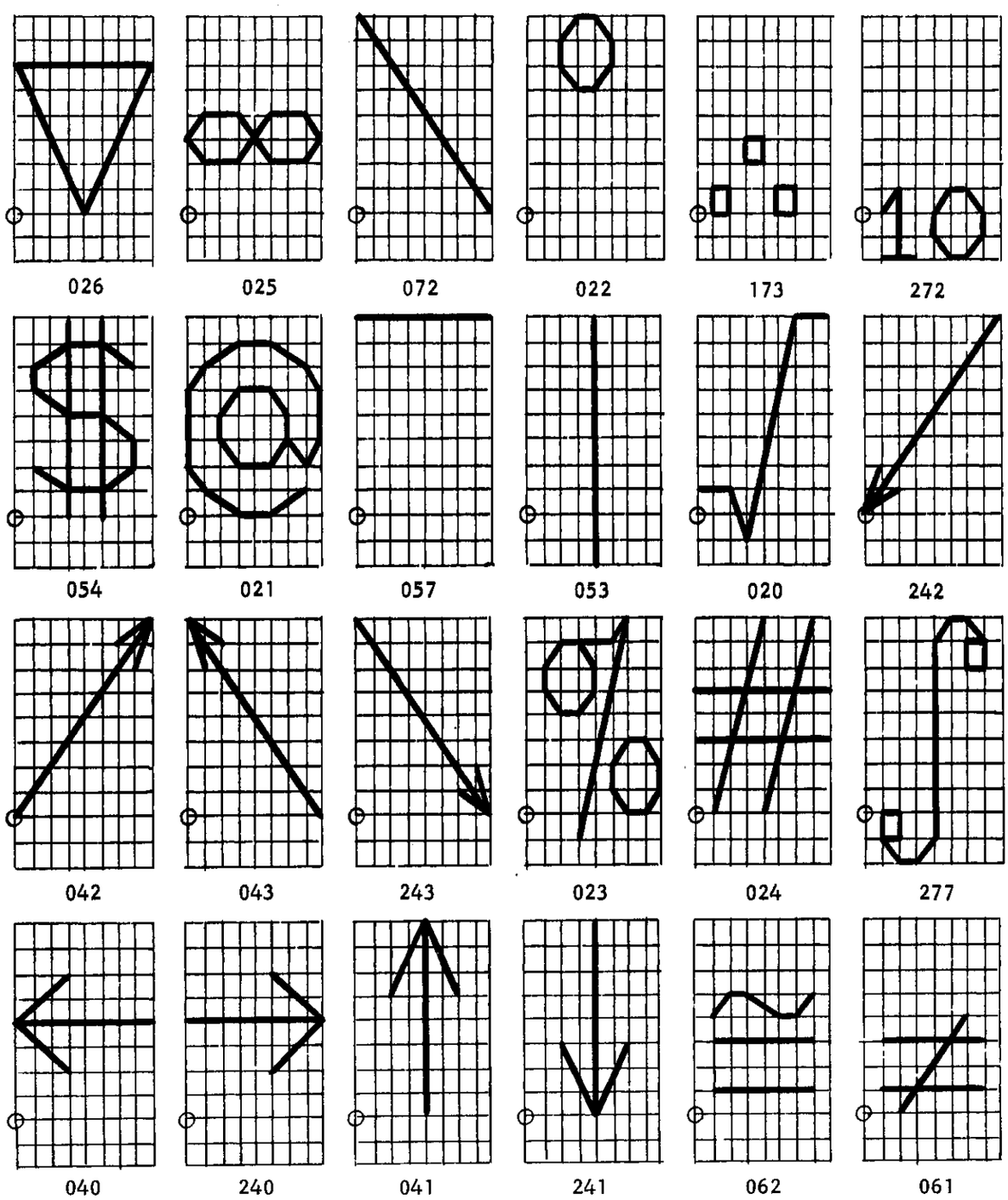


Figure A1-2 Character Generator Programs (cont'd)

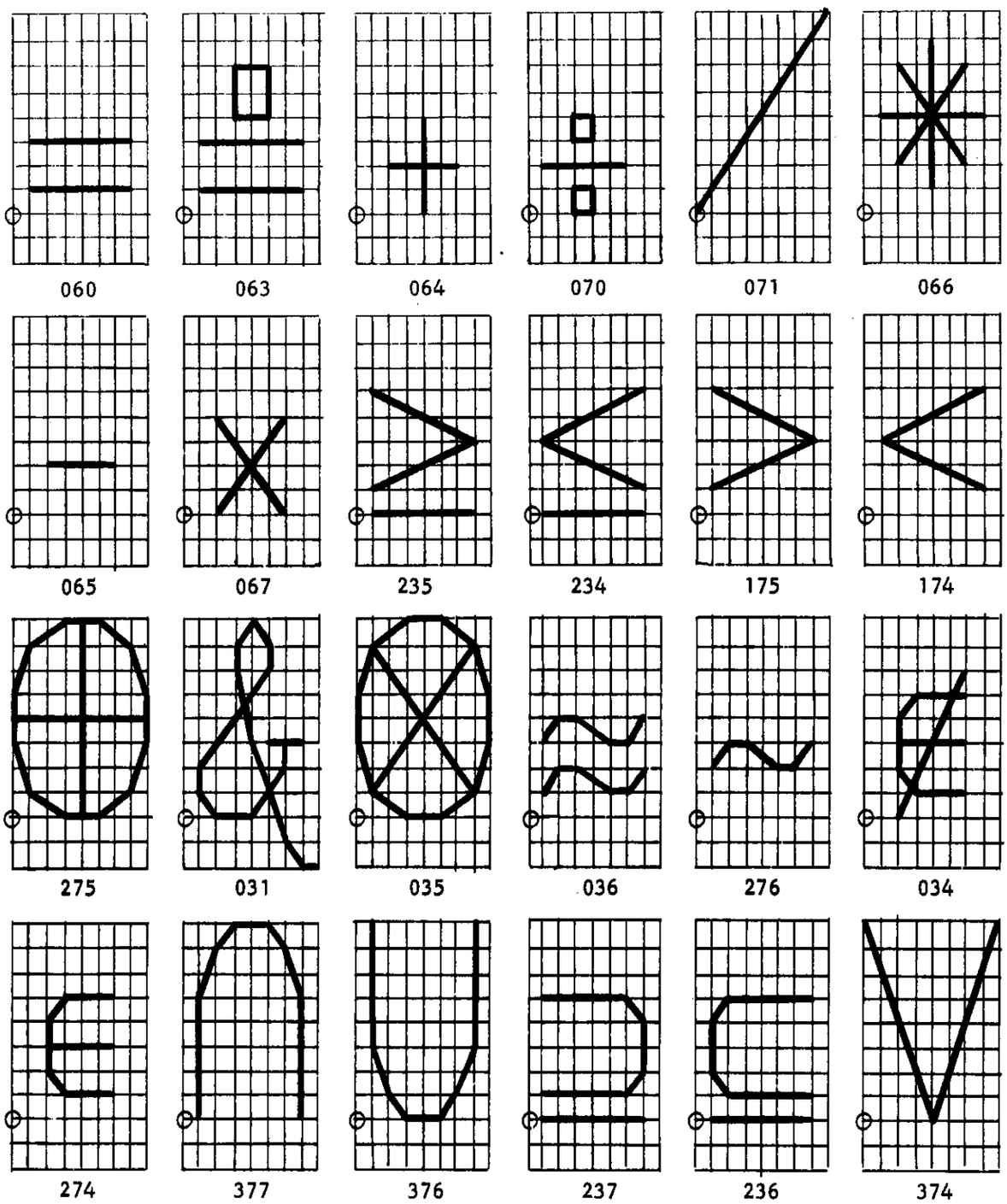


Figure A1-2 Character Generator Programs (cont'd)

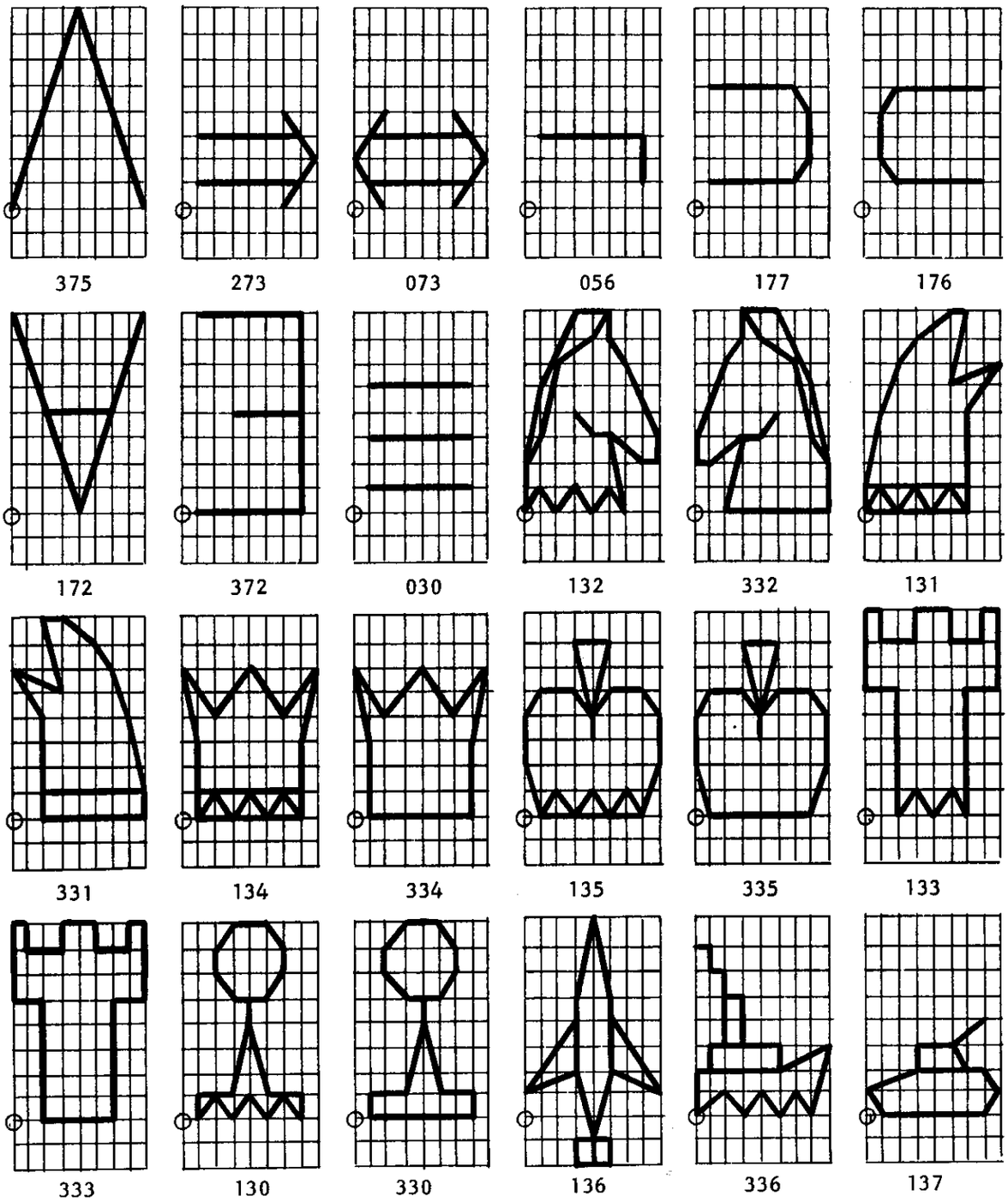


Figure A1-2 Character Generator Programs (cont'd)

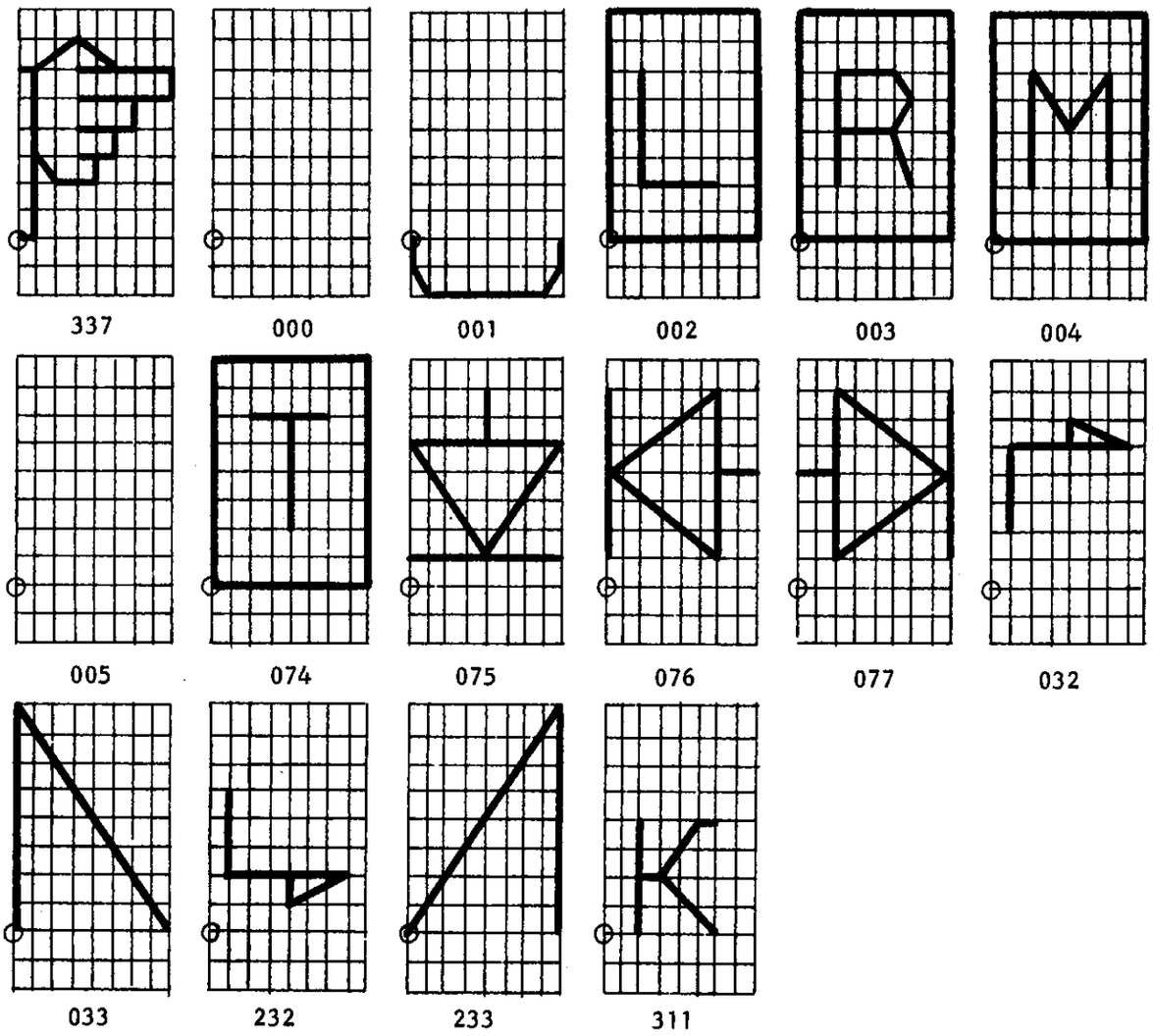


Figure A1-2 Character Generator Programs (cont'd)

APPENDIX 2 - THE VECTOR GENERATOR

The vector generator is manufactured by Philco for their READ display system. It is a transistorized control circuit which drives two D/A converters and a modulator. The outputs from the converters draw a straight line from the current beam position to the terminal point specified by ΔX and ΔY of the VECTOR STRING command. The line drawn deviates from an ideal straight line by no more than 2% of the line length. The terminal point is accurately positioned within 0.2% of the line length. The initial point, as specified by the format generator, is accurate within 0.1% of full scale.

After line drawing, the current beam position (major position) is incremented by the ΔX and ΔY values. The incrementation is performed digitally, by the format generator, so that drawing errors are not accumulative.

A logical adder within the vector generator approximates the length of line to be drawn. The time integrator which drives the converters uses the output of the logical adder to control drawing rates. All vectors are drawn at approximately the same rate so that they appear at the same intensity level.

The maximum permissible X and Y displacements are both 1024 (full screen). The maximum drawing time is the sum of set-up time and the line drawing time. The set-up time is constant at 2 μsec. The drawing time is approximately 7.5 μsec./in. Thus, at full scale deflection, the drawing maximum time is approximately 100 μsec.

APPENDIX 3 - THE FORMAT GENERATOR

The format generator is manufactured by Philco for their READ display system. It is a transistorized control circuit which drives two D/A converters. The converters define the major position for both the vector generator and character generator.

During character drawing, the format generator controls major positioning, automatic spacing, carriage returning, and margin setting. During vector drawing, the format generator controls major positioning, terminal point register updating, and general sequencing.

APPENDIX 4 - THE CONSOLE MAIN FRAME

The console main frame is shown in Figure A4-1. The CRT monitor, the state switches, the digital encoders, and the CRT controls are mounted on the front panel. The CRT controls are as follows.

X-centering: controls the X position of the entire display field.

Y-centering: controls the Y position of the entire display field.

focus : focuses the CRT beam.

intensity : adjusts the average beam intensity over the entire display field.

Character height : adjusts the height of all characters without affecting spacing.

Character width : adjusts the width of all characters without affecting spacing.

All units require 110-120 volts AC at 60 cps single phase. The display controller and all consoles must be connected to the same phase of the AC power line. The permissible ambient temperature range is 50° F to 75° F with a relative humidity of 80% or less.

The dimensions and cite requirements are shown in Figures A4-2, A4-3, and A4-4. Movable tables are provided for creating table top work space. A typical configuration is shown in Figure A4-5.

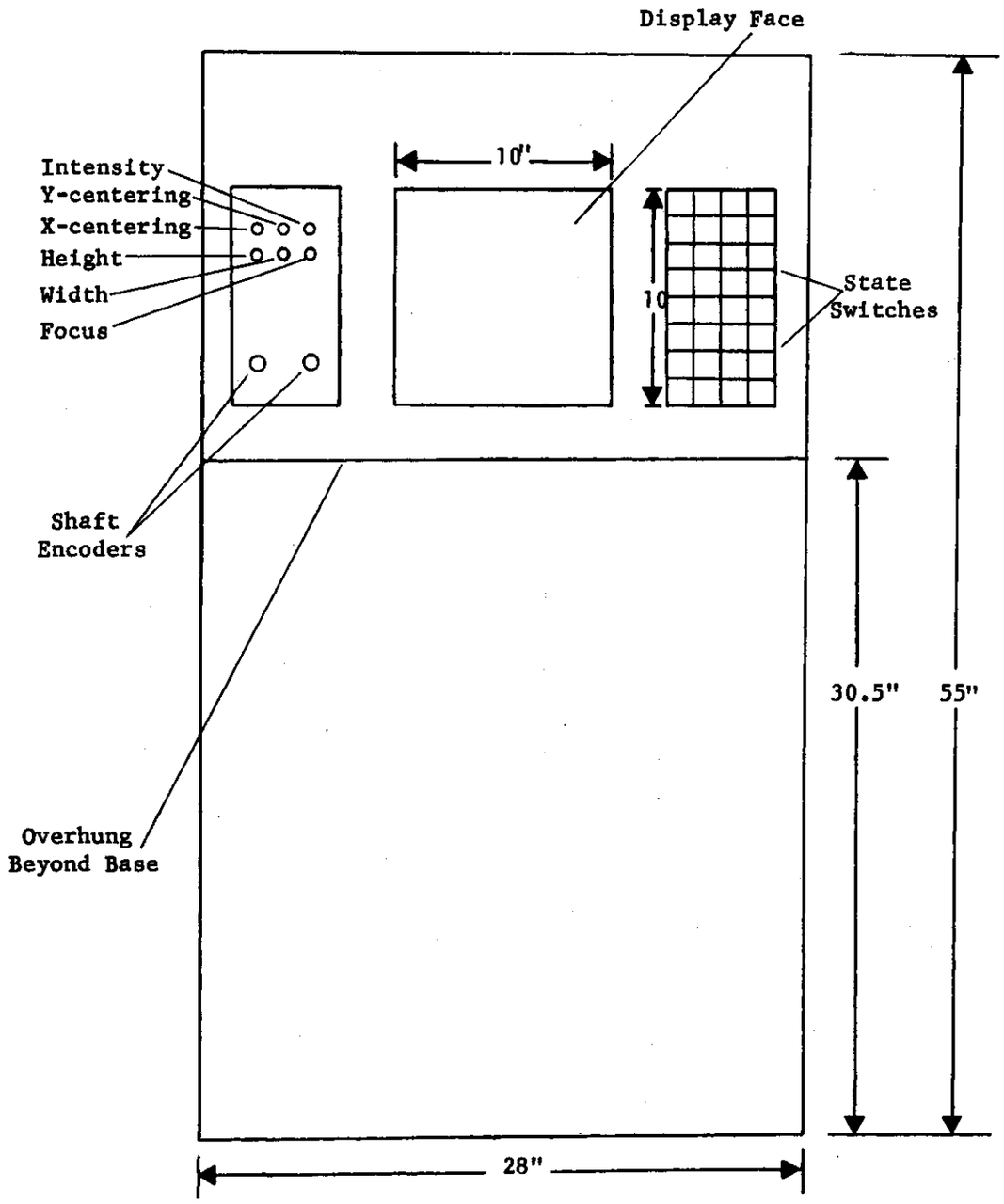


Figure A4-1 The Console Main Frame

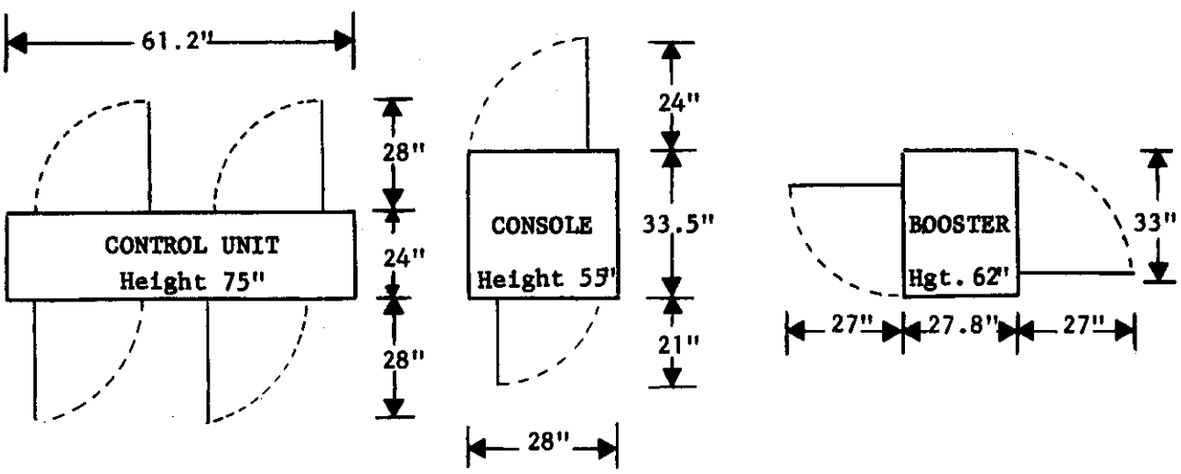


Figure A4-2 Unit Dimensions

UNIT	Heat Output BTU/Hour	Approx. Weight (Pounds)	Current Draw (Amps)	Service Outlet
Console	7,500	350	40A	None
Booster	500	100	10A	15A
Controller	4,500	800	20A	15A

Figure A4-3 Cite Requirements

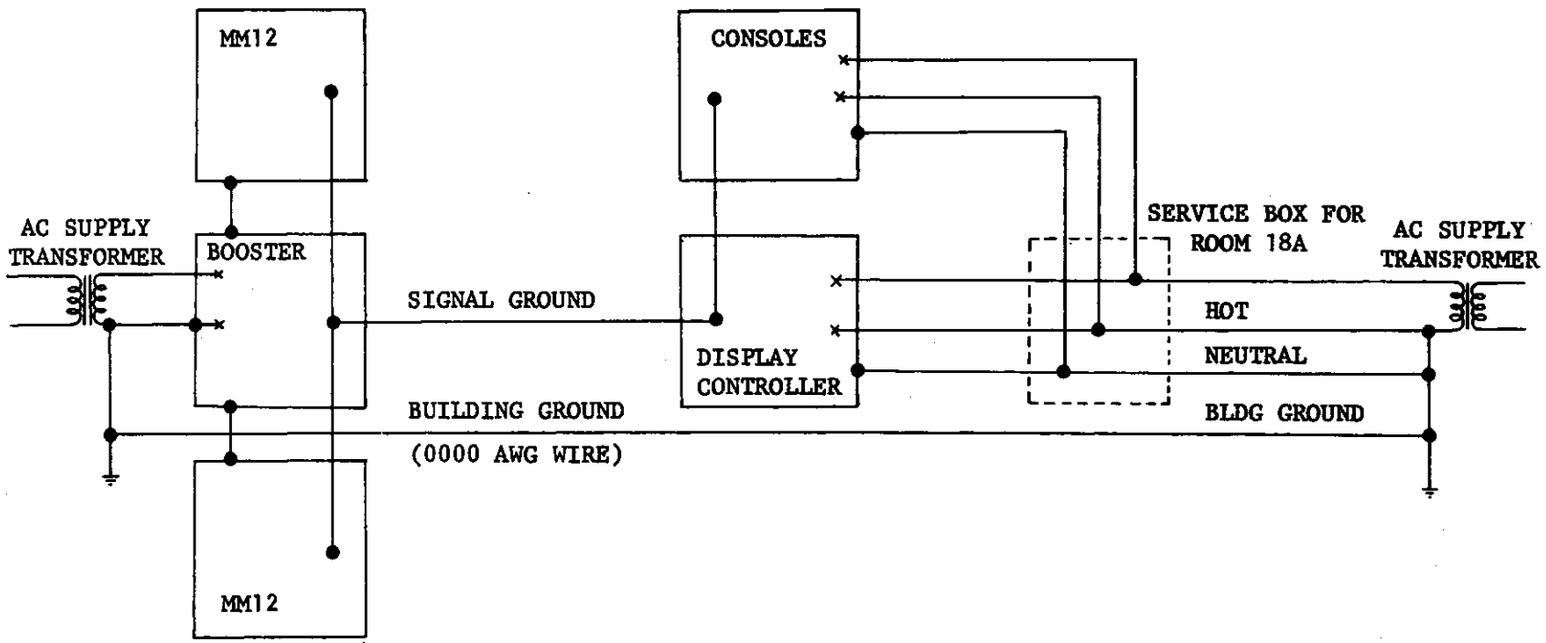


Figure A4-4 Grounding Requirements

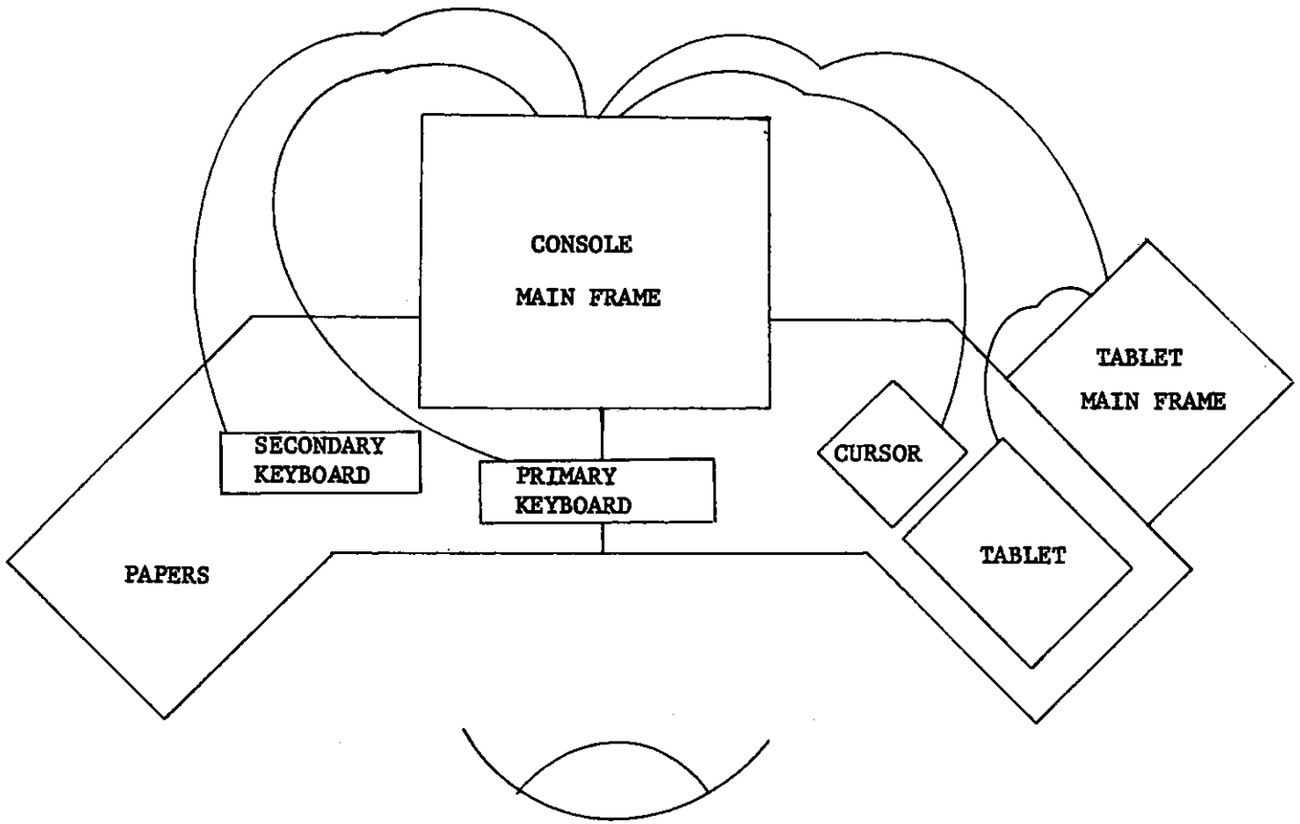


Figure A4-5 A Typical Console Configuration

APPENDIX 5 - THE CURSOR CONTROL AND INTERRUPT PANEL

The Cursor control and interrupt panel is a box housing 26 switches and connected to the main frame by means of a 10 foot cable. The switch configuration is shown in Figure A5-1.

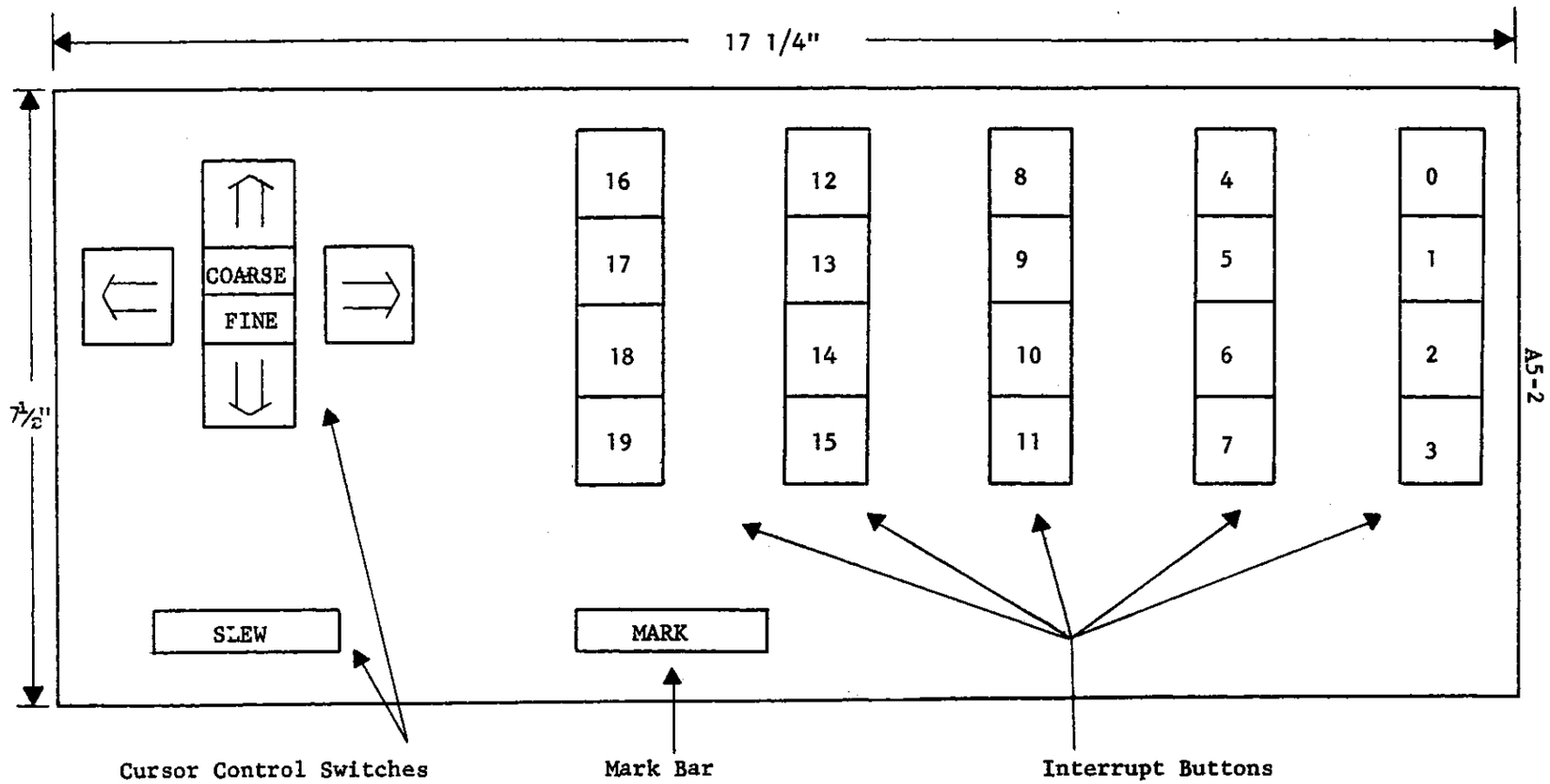


Figure A5-1 The Cursor Control and Interrupt Panel

APPENDIX 6 - THE TABLET

Appendix 6 will be issued at a later date.

APPENDIX 7 - THE KEYBOARDS AND CHARACTER SET

The primary and secondary keyboards are shown in Figures A7-1 and A7-2 respectively. An "underline" key is mounted on the primary keyboard. When underline is depressed, thirty keys (the Latin smalls, $<$, $>$, $\subset$, and $\supset$) produce codes for their underlined representations (the Latin smalls underlined, $\leq$, $\geq$, $\subseteq$, $\supseteq$). The shift key and the underline key are interlocked so that one must be released before the other is depressed. The full character set is shown in Figure A7-3. A summary of control characters is shown in Figure A7-4.

The code produced by the two keyboards assigns a unique 8-bit binary number to each character. The correspondence is shown in Figure A7-5. Bit positions are numbered from right to left starting at zero. The code transformations, under the operations of shift and underline, are given in Figure A7-6. The ith bit position of the character code is represented as C_i when $i = 0, 1, \ldots, 7$. The three transforms shown are labeled $UL(C_i)$ for underline, $PS(C_i)$ for primary keyboard shift, and $SS(C_i)$ for secondary shift. Only the most significant three bits are affected by any of the transforms. Thus, $T(C_i) = C_i$ where T = UL, PS, SS and i = 0, 1, 2, 3, 4.

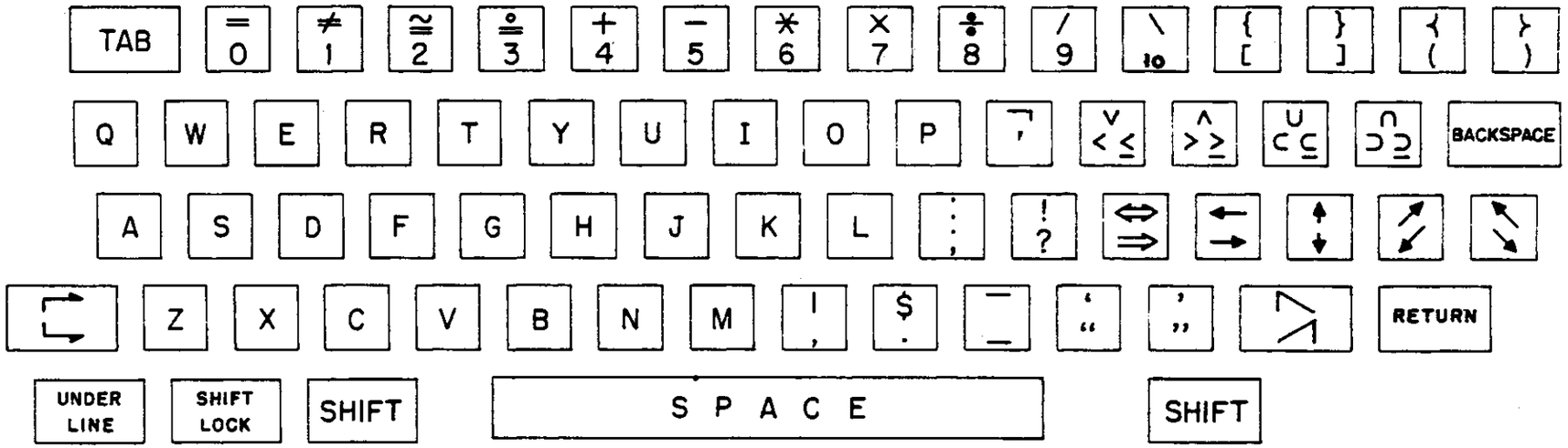
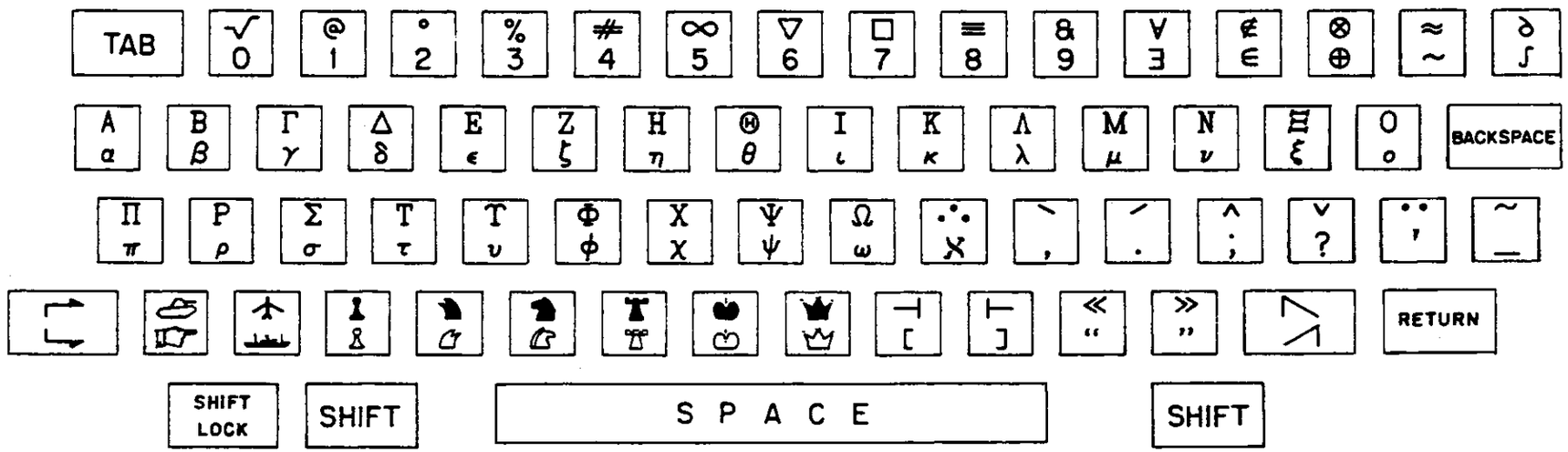
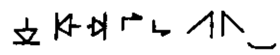


Figure A7-1 The Primary Keyboard



A7-2 The Secondary Keyboard

<u>Class</u>

CONTROL  [M] [L] [T] [R]

LATIN CAPS A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

LATIN SMALL a b c d e f g h i j k l m n o p q r s t u v w x y z

LATIN SMALL UNDERLINED <u>a</u> <u>b</u> <u>c</u> <u>d</u> <u>e</u> <u>f</u> <u>g</u> <u>h</u> <u>i</u> <u>j</u> <u>k</u> <u>l</u> <u>m</u> <u>n</u> <u>o</u> <u>p</u> <u>q</u> <u>r</u> <u>s</u> <u>t</u> <u>u</u> <u>v</u> <u>w</u> <u>x</u> <u>y</u> <u>z</u>

GREEK CAPS Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο Π Ρ Σ Τ Υ Φ Χ Ψ Ω

GREEK SMALL α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ υ φ χ ψ ω

DIGITS 0 1 2 3 4 5 6 7 8 9

PUNCTUATION . , ; : ' ! ? _ ‘‘ ’’ ‘ ’

DIACRITIC MARKS ´ ` ^ ˇ ~ ¨

PARENTHESES [ ] ( ) { } < > ≪ ≫ ⊣ ⊢

ARITHMETIC + - * × ÷ / = ≠ ≅ ≐ < > ≤ ≥

SET AND LOGIC ∧ ∨ ⊇ ⊆ ¬ ⇔ ⇒ ~ ≈ ≡ & ∃ ⊕ ⊗ ∈ ∉ ∀ ∩ ∪ ⊂ ⊃

MISCELLANEOUS MATH $_{10}$ $ ¯ | ↑ ↓ ← → \ ∫ √ # $ @ ° ∇ □ ∂ ∴ ↗ ↖ ↙ ↘ ✱ ∞

FREE SYMBOLS 

Figure A7-3 The Full Character Set

000		BLANK: Reserved code which neither displays nor spaces.
076	←	BACK SPACE: Decrements X coordinate register by 1 space.
077	→	FRONT SPACE: Increments X coordinate register by 1 space.
005		SPARE BLANK: Unused code which is interpreted as the BLANK character
002	L	LEFT MARGIN: Appears at left margin command position.
003	R	RIGHT MARGIN: Appears at right margin command position.
004	M	MARGIN RELEASE: Inhibits margin control until either X ≥ 1023 or encountered.
075	↵	CAR RET and LINE FEED: Combines the two operations into one character.
074	T	TAB: Acts like Space
032	↱	SUPER: Positively biases by 1/2 line.
232	↳	SUB: Negatively biases by 1/2 line.
233	/	90/LINE: Modifies size of subsequent characters to scale 90 CHAR/LINE (independent of size bit).
033	\	NORMAL: Return size interpretations to normal.
001	_	CURSOR: Reserved code used by display to generate cursor symbol.

Figure A7-4 The Control Characters

000	040 ←	100 A	140 a	200 a	240 →	300 α	340 A
001 ⌣	041 ↑	101 P	141 b	201 b	241 ↓	301 β	341 B
002 [L]	042 ↗	102 Γ	142 c	202 c	242 ↙	302 γ	342 C
003 [R]	043 ↖	103 Δ	143 d	203 d	243 ↘	303 δ	343 D
004 [M]	044 ↓	104 E	144 e	204 e	244 (	304 ε	344 E
005	045 ↑	105 Z	145 f	205 f	245)	305 ζ	345 F
006 ⊣	046 {	106 H	146 g	206 g	246 [	306 η	346 G
007 ⊢	047 }	107 O	147 h	207 h	247]	307 Θ	347 H
010 ≪	050 ‘	110 I	150 i	210 i	250 “	310 ι	350 I
011 ≫	051 ’	111 K	151 j	211 j	251 ”	311 k	351 J
012 ^	052 :	112 Λ	152 k	212 k	252 ;	312 λ	352 K
013 ↖	053 |	113 M	153 l	213 l	253 ,	313 μ	353 L
014 ↗	054 $	114 N	154 m	214 m	254 .	314 ν	354 M
015 ∨	055 !	115 Ξ	155 n	215 n	255 ?	315 ξ	355 N
016 ••	056 ¬	116 O	156 o	216 o	256 '	316 ο	356 O
017 ~	057 ¯	117 Π	157 p	217 p	257 _	317 π	357 P
020 √	060 =	120 P	160 q	220 q	260 0	320 ρ	360 Q
021 @	061 ≠	121 Σ	161 r	221 r	261 1	321 σ	361 R
022 °	062 ≅	122 T	162 s	222 s	262 2	322 τ	362 S
023 %	063 ≙	123 Y	163 t	223 t	263 3	323 υ	363 T
024 #	064 +	124 Φ	164 u	224 u	264 4	324 φ	364 U
025 ∞	065 -	125 X	165 v	225 v	265 5	325 χ	365 V
026 ▽	066 *	126 Ψ	166 w	226 w	266 6	326 ψ	366 W
027 □	067 ×	127 Ω	167 x	227 x	267 7	327 ω	367 X
030 ≡	070 ÷	130	170 y	230 y	270 8	330	370 Y
031 &	071 /	131	171 z	231 z	271 9	331	371 Z
032	072 \	132	172 ∀	232	272 10	332	372
033	073 ⇔	133	173 ∴	233	273 ⇒	333	373
034 ∉	074 [T]	134	174 <	234 ≤	274 ∈	334	374 ∨
035 ⊗	075	135	175 >	235 ≥	275 ⊕	335	375 ∧
036 ≈	076	136	176 ⊂	236 ⊆	276 ~	336	376 ∪
037 ∂	077	137	177 ⊃	237 ⊇	277 ∫	337	377 ∩

Figure A7-5 The Character Code

SECONDARY SHIFT (SECONDARY KEYBOARD)	$SS(C7) = 0$
	$SS(C6) = C6$
	$SS(C5) = \overline{C7}$
PRIMARY SHIFT (PRIMARY KEYBOARD)	$PS(C7) = C6$
	$PS(C6) = C6$
	$PS(C5) = C5$
UNDERLINE (PRIMARY KEYBOARD)	$UL(C7) = C7 \vee C6$
	$UL(C6) = 0$
	$UL(C5) = C5 \wedge \overline{C6}$

	C7	C6	C5
SS:	0	C6	$\overline{C7}$
PS:	C6	C6	C5
UL:	$C7 \vee C6$	0	$C5 \wedge \overline{C6}$

Figure A7-6 The Code Transformations