REVISED AS OF 28 SEP 82. →

82-09-3

ENG -

T500
X3L2/82-96

ANSI Draft Standard CSA

Videotex/Teletext Presentation Level Protocol Syntax (North American PLPS)

sept 28/82

~~1 August 1982~~

Prepared Jointly by ANSI X3L2.1 Videotex/Teletext
Character Sets Standing Task Group and
CVCC/CSA/WG on Videotex

Contents

Videotex/Teletext Presentation Level Protocol Syntax
(North American PLPS)

1. Scope

1.1 This Standard describes the formats, rules, and procedures for the encoding of alphanumeric text and pictorial information for videotex and teletext* applications. This Standard conforms to the architecture defined in ISO's multi-layered reference model of open systems interconnection as part of a presentation level protocol. A presentation level protocol is one of seven protocol specifications that would be required to completely define a videotex or teletext standard.

The basic coding scheme is built upon the framework established by the CCITT Recommendation S.100, International Information Exchange for Interactive Videotex. Operation in both a 7-bit and an 8-bit environment is accommodated. Alphanumeric text, a set of Supplementary Characters, and the Dynamically Redefinable Character Set (DRCS) are provided. Both mosaics and geometric primitives, as well as DRCS, can be used to create pictorial displays. The mosaic coding is compatible with CCITT Recommendation S.100. The geometric primitives are compatible enhancements to the Picture Description Instructions (PDIs) defined in the alphageometric option of CCITT Recommendation S.100. Additional capabilities include color mapping, a controllable stroke width, macros, continuous character scaling, programmable texture masks, unprotected fields, partial screen scrolling, and incremental encoding for highly compact descriptions of certain classes of images.

**The term "teletext" (not yet adopted by CCIR) is commonly used to refer to a broadcast television videotex service. It is different from "teletex", a term officially adopted by CCITT to define a specific type of terminal-to-terminal text communications service. This is an area of confusion that has been recognized by CCITT and CCIR.*

2. Definitions

2.1 The following definitions apply in this Standard:

Absolute coordinates means an ordered pair or triplet of signed numbers between -1 (inclusive) and 1 (non-inclusive) that specifies (in two's complement arithmetic) the new location of the drawing point with respect to the origin of the unit screen. Only positive absolute coordinate specifications lie within the unit screen;

Alphanumeric text is the written form of languages and comprises alphabetic letters with or without diacritical marks, numerical digits and fractions, punctuation marks, typographical symbols, and mathematical signs as well as SPACE and special letters, signs, symbols, etc. In this Standard, alphanumeric text characters are denoted by names that are intended to reflect their customary meanings for the characters and symbols displayed. These names are not intended to specify a particular style, font design, character size, or position within a character field;

Attribute means a settable parameter to be applied to subsequent alphanumeric text or pictorial information;

Bit combination is an ordered set of bits (binary digits) that represents a character or a control function;

Border area means the area of the physical display screen that is outside the physical display area;

C-set stands for control set. There are two control sets, C0 and C1, each of which comprises 32 character positions arranged in 2 columns by 16 rows;

Character field means the rectangular area within which a character is displayed;

Code extension means techniques for expanding the absolute character address space of a byte-oriented code into a larger virtual address space;

Code table means the set of unambiguous rules that defines the mapping between received bit combinations and presentation level characters;

Coding interface is an interface through which coded bit combinations are passed between terminal equipment and communication media;

Color map address means an ordinal number associated with each pixel in a stored digital image that determines the address in the color map at which the actual color value of that pixel can be found. (This is sometimes abbreviated to simply "color" when it can be done unambiguously.);

Color map means a look-up table that is used during scan conversion of the digital image that converts color map addresses into actual color values;

Color value is an entry in a color map that indicates the actual color of the pixel to be displayed;

Composite symbol is a symbol consisting of a combination of two or more symbols in a single character field, such as a diacritical mark and a basic letter;

Cursor is a logical indicator of the screen position at which the next character is to be deposited. This position may or may not be marked by a cursor symbol;

Dynamically Redefinable Character Set (DRCS) is a G-set containing definable characters whose patterns can be downloaded from the host;

Designate means to identify a given set from the repertory of G-sets as a G0, G1, G2, or G3 set;

Drawing point is a logical indicator of the screen position at which the next geometric graphic primitive will commence execution. This is not normally marked by a drawing point symbol;

Escape sequence means a string of two or more bit combinations beginning with the ESC character. A three-character escape sequence contains an intermediate character (I) and ends with a final character (F), and is used primarily to designate a set of 94 or 96 character codes as one of the 4 active G-sets. Two-character escape sequences contain only a final character (F) and are one method by which code sets are invoked into the in-use table. Formats and rules regarding the use of the escape sequences are specified in ISO DIS 2022.2;

Final character is the last character of an escape sequence;

G-set refers to one of the four sets, G0, G1, G2, and G3, each of which comprises 94 or 96 character positions arranged in 6 columns by 16 rows;

G-set repertory is the collection of available code sets that are subject to designation as one of the G-sets;

Geometric graphic primitive is a locally stored picture drawing algorithm that can be called via a specified opcode and associated operand(s);

Graphic character repertoire is the list of graphic characters defined in this Standard, including accented letters and characters obtained by the composition of two or more graphic symbols;

Implementation dependent means that the feature may be specified more completely in a Service Reference Model or by an implementor, within the constraints imposed by this Standard.

In-use refers to the code sets or attributes that will be used to interpret or be applied to subsequently received commands;

Intermediate character is any character that occurs between the escape character and the final character in an escape sequence;

Invoke means to cause a designated code set to be represented by the bit combinations in the prescribed in-use table;

Layer is terminology adopted by ISO to describe each individual module of the reference model for open system interconnection (OSI). (The terms "level" and "layer" are used interchangeably.)

Locking shift means an invocation of a code set into the in-use table that remains in effect until another code set is invoked in its place;

Logical picture element (logical pel) is a geometric construct associated with the drawing point whose size determines the stroke width of graphics primitives;
Note: Although the terms pixel and pel are synonymous in common usage, throughout this document pixel is used for physical picture elements and pel for logical picture elements.

Macro is a locally stored string of presentation code represented with a single-character name. When the macro-name is used the locally stored string is processed in its place;

Mosaic is a rectangular matrix of pre-defined elements that can be used to construct block-style graphic images;

Opcode is a one-byte, presentation-level character that initiates the execution of a locally stored geometric primitive or control operation. An opcode may be followed by one or more operands;

Operand is a single- or multiple-byte string from the numeric data field of the PDI code set that is used to specify control, attribute, or coordinate parameters required by the opcode;

Physical display area means the addressable area of the physical display screen onto which the unit screen or a portion of the unit screen is mapped;

Physical picture element (pixel) is the smallest displayable unit on a given display device;

Pictorial information is the display information resulting from the application of geometric primitives, mosaics, and DRCS;

Nominal black is the color black (all zeroes) in color mode zero, or the color that is at color map address zero in color modes 1 and 2.

Picture description instruction (PDI) is composed of an opcode followed by ~~one~~ or more operands and constitutes an executable picture drawing or control command;

Presentation level is the sixth of seven protocol levels defined by ISO's reference model of open systems interconnection. The presentation level (or layer) is primarily responsible for the encoding of text, graphic, and display control information;

Protocol is a set of formats, rules, and procedures governing the exchange of information between peer processes (levels);

Relative coordinates is an ordered pair or triplet of signed numbers between -1 (inclusive) and 1 (non-inclusive) that specifies (in two's complement arithmetic) either the new location of the drawing point with respect to the old location of the drawing point, when used within a geometric primitive, or the dimensions of a given field when used with one of the control commands;

Single shift is an invocation of a code set into the in-use table that affects only the interpretation of the next bit combination received. Interpretation then automatically reverts to the previous contents of the table. (This is also referred to as non-locking shift.)

Terminal equipment is equipment that can exchange coded bit combinations by means of telecommunication or by physical interchange of storage media;

Unit screen means the logical display address space within which all drawing operations are executed and alphanumeric characters are deposited. The dimensions of the unit screen are 0 to 1 in the horizontal (X), vertical (Y), and depth (Z) dimensions. (The last is only defined in three-dimensional mode.)

Service Reference Model (SRM) defines the minimum set of features that must be implemented by a terminal or decoder for a particular service an the maximum set of features that may be us to create information.

3. Reference Publications

3.1 This Standard refers to the following Publications and where reference is made it shall be to the edition listed below, including all revisions published thereto:

ANSI* Standards
X3.4-1977,
American National Standard Code for Information Interchange;

X3.41-1974,
American National Standard Code Extension Techniques for Use with the 7-Bit Coded Character Set of American National Standard Code for Information Interchange;

CSA† Standards
Z243.4-1973,
7-Bit Coded Character Sets for Information Processing Interchange;

Z243.35-1976,
Code Extension Techniques for Use with the 7-Bit Coded Character Sets of CSA Standard Z243.4-1973;

CCIR‡ Report
957-October, 1981,
Characteristics of Teletext Systems, Document 11/5001-E;

CCITT§ Recommendations
S.100-1980,
International Information Exchange for Interactive Videotex;

F.300-1980,
Videotex Service;

Department of Communications, Canada
Telecommunications Regulatory Service,
Broadcast Specification BS-14 June, 1981

ISO** Standards
DIS 646-March, 1982,
Data Processing - 7-Bit Coded Character Set for Information Interchange;

2022-1973,
Code Extension Techniques for Use with the ISO 7-Bit Coded Character Set;

2022.2-February, 1982,
Information Processing - ISO 7-Bit and 8-Bit Coded Character Sets - Code Extension Techniques;

DIS 6937/1-February, 1982,
Information Processing - Coded Character Sets for Text Communication - Part 1: General Introduction;

DIS 6937/2-February, 1982,
Information Processing - Coded Character Sets for Text Communication - Part 2: Latin Alphabetic and Non-Alphabetic Graphic Characters;

DP 7498-February, 1982,
Data Processing - Open Systems Interconnection Basic Reference Model.

*American National Standards Institute.

†Canadian Standards Association

‡International Radio Consultative Committee.

§International Telegraph and Telephone Consultative Committee.

**International Organization for Standardization.

Note: DP = Draft Proposal; DIS = Draft International Standard.

4. Coding Architecture

4.1 Reference Model (OSI). The coding scheme described in this Standard addresses itself to the presentation level of the ISO seven layer reference model for open systems interconnection. This reference model is described in ISO DP 7498, Data Processing - Open Systems Interconnection Basic Reference Model. Table 1 illustrates the model. The ISO layered system architecture is an assembly of interrelated protocols required to define an entire communications system. The layered system architecture allows an "open systems interconnection" (i.e., data exchange) between participating systems. Each layer covers an independent aspect of a communications system in such a way that other protocols may be substituted at various layers in order to operate over different media. Both videotex and teletext services make use of the same presentation level protocol but may have different protocols at other layers. The protocols for layers 1 to 5 for teletext are defined in CCIR, Report 957, "Characteristics of Teletext Systems", Document 11/5001-E, October 1981; and Broadcast Specification BS-14, Department of Communications, Canada, Telecommunications Regulatory Service, June 1981.

Table 1

The Seven Functionally Separate Layers of the OSI Model

Function	
	END USER APPLICATION PROCESS
Provides appropriate service for application	7 Application
Provides data formatting	6 Presentation
Provides service facilities to the application	5 Session
Provides for end-to-end data transmission integrity	4 Transport
Switches and routes information units	3 Network
Provides transfer functions for units of information to the other end of the physical link	2 Data Link
Transmits bit stream to physical medium	1 Physical

Note: The seven layers may be viewed in two major groupings. Layers 1 to 4 concern the transference of data while layers 5 to 7 concern how the data is processed and used.

(1) The Physical Layer provides mechanical, electrical and procedural functions in order to establish, maintain, and release physical connections.

(2) The Data Link Layer provides a data transmission link across one or several physical connections. Error correction, sequencing, and flow control are performed in order to maintain data integrity.

(3) The Network Layer provides routing, switching, and network access considerations in order to make invisible to the transport layer how underlying transmission resources are utilized.

(4) The Transport Layer provides an end-to-end transparent virtual data circuit over one or several tandem network transmission facilities.

(5) The Session Layer provides the means to establish a session connection and to support the orderly exchange of data and other related control functions for a particular communication service.

(6) The Presentation Layer provides the means to represent and interpret the information in a data coding format in a way that preserves its meaning. The detailed coding formats for the scheme described in this document provide the basis of a Presentation Level Protocol for videotex, teletext, and related applications.

(7) The Application Layer is the highest layer in the reference model and the protocols of this layer provide the actual service sought by the end user. As an example, the information retrieval service commands of a videotex application form part of the application layer protocol.

4.2 Presentation Level Overview

4.2.1 General. This Presentation Level Protocol Syntax is structured based on the code extension principles of ISO DIS 2022.2 and on currently existing national and international standards and recommendations. This clause describes the relationship of the various coding standards and the manner in which they are woven into a unified presentation level protocol syntax.

In the character coded method of describing alphanumeric characters and pictorial information particular character codes are identified by an 8-bit code sequence in which 7 of the bits are used as an index into a 128 character code table and the eighth bit is used for extension to another code table of 128 characters, as will be described later in this Standard or for use at other protocol layers, for example, parity.

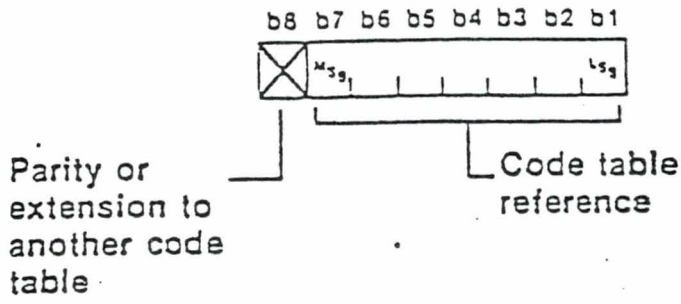


Figure 1

Coding Format

The character code table is normally represented as a table of 8 columns and 16 rows with bits b7, b6, and b5 addressing the columns and bits b4, b3, b2, and b1 addressing the rows (see Figure 3). This general format is used throughout this Standard. In diagrams, the bits are numbered b1 to b8 with b1 occupying the least significant position. See Figure 1. The code table may be sub-divided into different segments as detailed in Clause 4.3.

The coding of alphanumeric characters is based on current national standards (ANSI X3.4 "American National Standard Code for Information Interchange" and CSA Standard Z243.4, 7-Bit Coded Character Sets for Information Processing Interchange) and international standard (ISO DIS 646, Data Processing - ISO 7-Bit and 8-Bit Coded Character Sets for Information Interchange) as the Primary Character Set (see Clause 5.1) together with a Supplementary Character Set based on CCITT S.100 (see Clause 5.2).

The coding of the pictorial information is based on:

(a) Picture Description Instructions (PDI) (see Clause 5.3), which are based on an enhancement of the alphageometric option described in CCITT Recommendations S.100 and F.300;

(b) Mosaic Set (see Clause 5.4), which is based on the union of the two mosaic tables described in CCITT Recommendations S.100 and F.300;

(c) Macro Set (see Clause 5.5);

(d) Dynamically Redefinable Character Set (DRCS) (see Clause 5.6).

Independence of display hardware constraints is achieved by using simple geometric picture description instructions as the basis of the coding scheme. The geometric drawing commands are defined in terms of an abstract unitary coordinate space (i.e., unit screen), and the text oriented commands are defined in terms of a variable character size that permits different size characters to be displayed on the screen.

4.2.2 Coordinate System. The coordinate system utilized is based on the concept of the unit screen. Execution of the geometric primitives always occurs within the unit screen, whose horizontal (X), vertical (Y), and depth (Z)* coordinates range from 0 to 1. The unit screen is mapped to the physical display screen such that the origin (0,0) of the unit screen is mapped to the lower left corner of the display. On devices with a square display screen, the upper left corner of the unit screen (0,1) is mapped to the upper left corner of the display. For example, on devices with a rectangular display screen, in particular the cathode ray tubes found in television sets, which have an approximately 4:3 aspect ratio (see Figure 2), the upper left corner of the unit screen is mapped outside the display area such that the upper left corner of the display actually corresponds to a Y coordinate, within the unit screen, of approximately 0.75. (This number may vary depending on the particular implementation, most importantly on the number of scan lines included in the display.)

**The depth (Z) coordinate specification is only meaningful on terminals with a capability for operating in three-dimensional mode. It is allowed for in the data structures both for logical completeness and to facilitate the graceful introduction of this feature when the technology becomes available. Note that a Z value of 0 is interpreted as being farthest from the viewer, i.e., lowest order plane. In general, two-dimensional mode will be assumed, with complete specification of three-dimensional mode operation deferred for future standardization.*

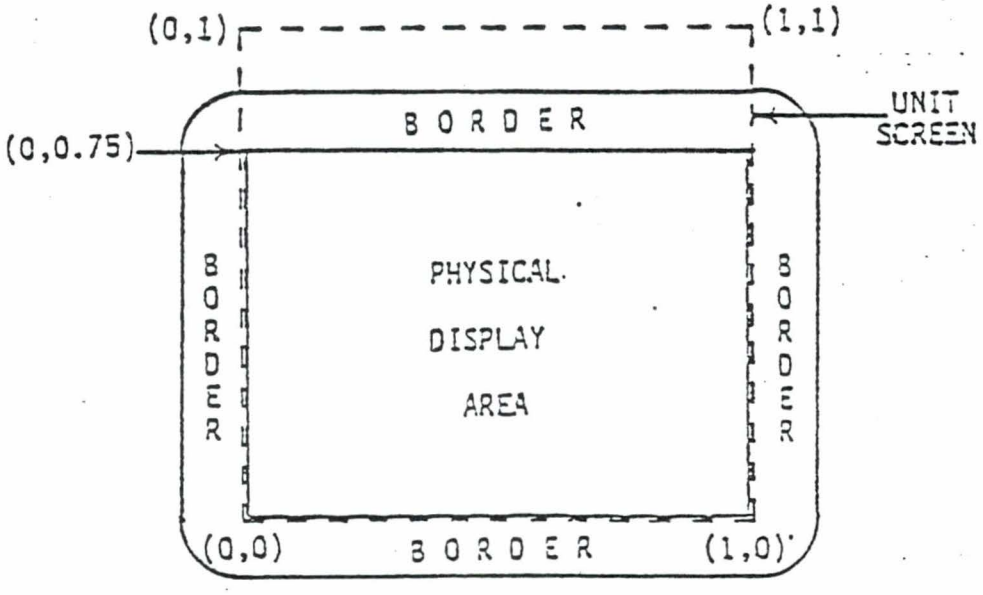


Figure 2

Unit Screen Concept

Note: The sender can always assume that the entire physical display area is available. It is the responsibility of the terminal manufacturer to ensure that the entire physical display area lies within the "safe displayable area" depending on the implementation.

The coordinate data defined by the PDI operands can be interpreted either as the absolute coordinates within the unit screen of a logical drawing point or as displacements from the previous drawing point, depending on the context defined by the particular opcode. This drawing point is then used in the execution of the geometric primitives described in Clause 5.3.3.

4.2.3 Display Format. There is no special positional dependence upon the order in which drawing primitives are presented. Pictures are built up out of a sequence of drawing commands, with each superimposed over previous ones. In this manner pictures are built up in layers. Specifically, this means that composite pictorial or alphabetic character images may be composed by the superimposition of multiple characters and/or drawing primitives.

The registration between alphanumeric characters and pictures is not guaranteed to be exact, nor is the superimposition between these drawing modes guaranteed to be exact. This permits this Standard to be implemented using various types of display hardware. For the display of alphanumeric characters in the absence of any specific character size specification (either by the TEXT opcode or the C1 controls) the display format will be 40 characters by 20 rows. Other display formats are permitted when the appropriate commands are used to define the suitable character sizes (see Appendix A).

4.3 Code Extension

4.3.1 General

4.3.1.1 This Clause describes the method of code extension to be used. Such a method is based on the code extension techniques specified in ISO 2022 (the corresponding ANSI and CSA Standards are ANSI X3.41 and CSA Z243.35, respectively).

It provides the capability to "designate" from the repertory of sets and "invoke" into the in-use table, where a specified byte of coded data acts as a pointer into a combined code table consisting of C- and G-sets. In most applications there are not enough characters available in the in-use table, so provision has been made in the structure to permit G- or C-sets to be switched.

4.3.1.2 The entire coding environment described in this Standard is to be designated and invoked as a "complete code" by the escape sequence ESC 2/5 F_1, in accordance with ISO 2022, where F_1 is the final character to be registered with ISO. Conforming interchange does not require the use of

this escape sequence except when interchanging with other services. In this complete code, special attention is to be paid to the following:

G0: A 94 code position G-set
G1: A 94 or 96 code position G-set
G2: A 94 or 96 code position G-set
G3: A 94 or 96 code position G-set

A 94 code position G-set is one that does not include code positions 2/0 and 7/15. When such a set is designated as G0, these two positions shall have the meanings of SPACE and DELETE respectively.

A 96 code position G-set, on the other hand, is one in which the positions 2/0 and 7/15 have meanings other than SPACE and DELETE.

The designation and invocation of this "complete code" will be terminated by a different sequence ESC 2/5 F_2 (to be registered or standardized by ISO) or by the designation and invocation of any other "complete code".

4.3.1.3 There are four G-sets and two C-sets that are designated at any one time; that is, any one of the four tables G0, G1, G2 or G3 could be invoked into the in-use table by an invocation sequence. Invocation sequences may be either locking or non-locking. The G0, G1, G2, and G3 sets act as slots into which code sets from the G-set repertory of meanings may be designated. In the default state G0 contains the Primary Character Set, G1 the PDI Set, G2 the Supplementary Character Set and G3 the Mosaic Set. A designation sequence is used to establish a new meaning to a code set slot.

4.3.1.4 The choice of the 7-bit or 8-bit code environment at the presentation level may be established by other protocol layers for a particular service, or by prior agreement.

The in-use table is structured into 32 code position C-sets and 94 or 96 code position G-sets. The contents of these sets apply to either the 7-bit or the 8-bit environment. These sets are manipulated for the purpose of providing a virtual address space larger than the 128 or 256 code positions available in a 7-bit or 8-bit environment respectively.

4.3.2 Code Extension in a 7-Bit Environment. A 128 code position in-use table is defined, as shown in Figure 3. Each incoming bit combination is either decoded according to the current contents of this table or is used to change the contents of this table. The table itself is organized into 8 columns of 16 rows, with bits 1 through 4 defining the row number and bits 5 through 7 defining the column number. The in-use table contains, in columns 0 and 1, the C0 set. Five characters of this set, ESCAPE (ESC or 1/11, i.e., column 1, row 11), SHIFT IN (SI or 0/15), SHIFT OUT (SO or 0/14), SINGLE SHIFT TWO (SS2 or 1/9), and SINGLE SHIFT THREE (SS3 or 1/13), are used to control the contents of the remaining 6 columns of the in-use table. The manner in which this is accomplished is graphically depicted in Figure 4 and is described below.

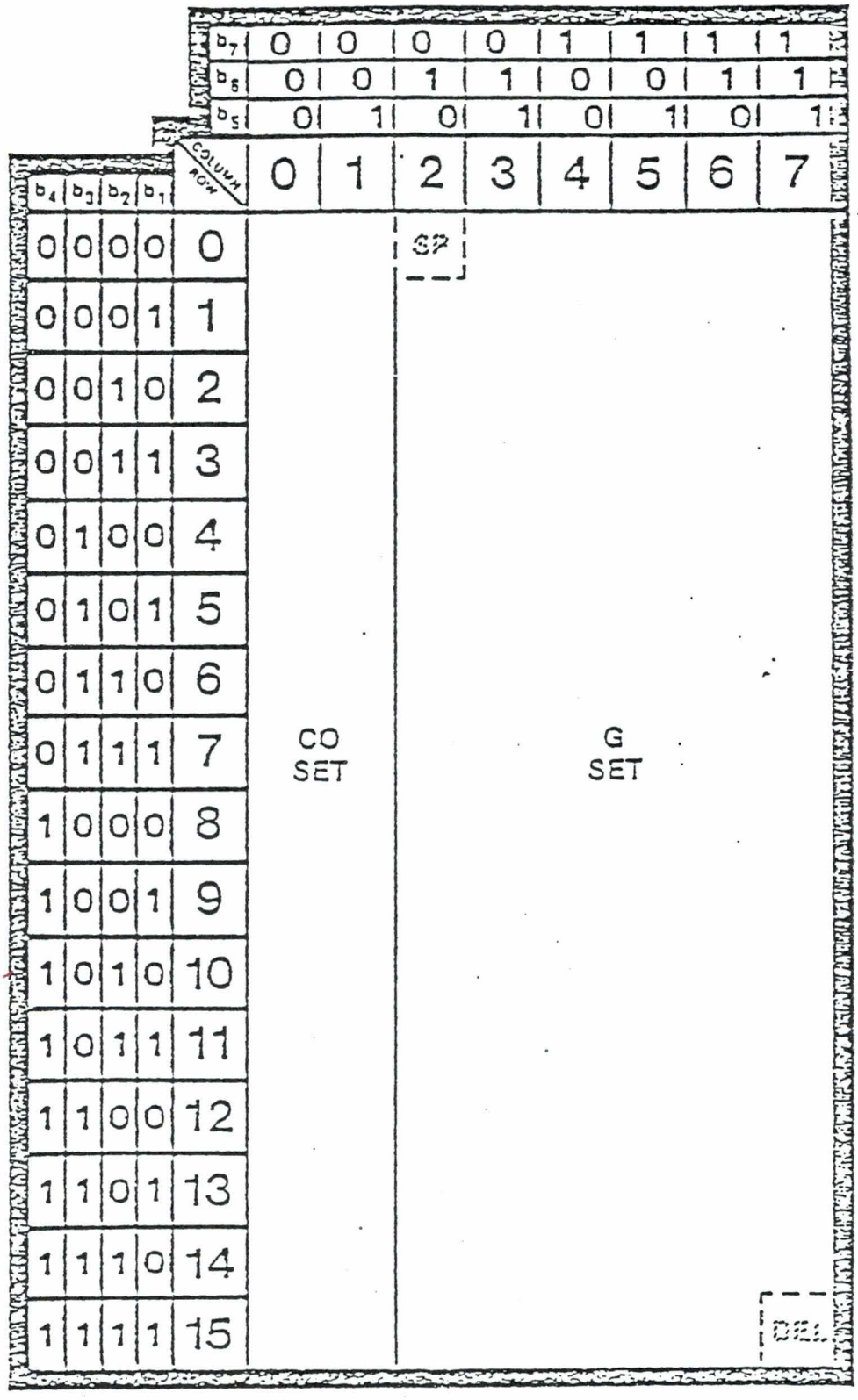


Figure 3

7-Bit In-Use Table

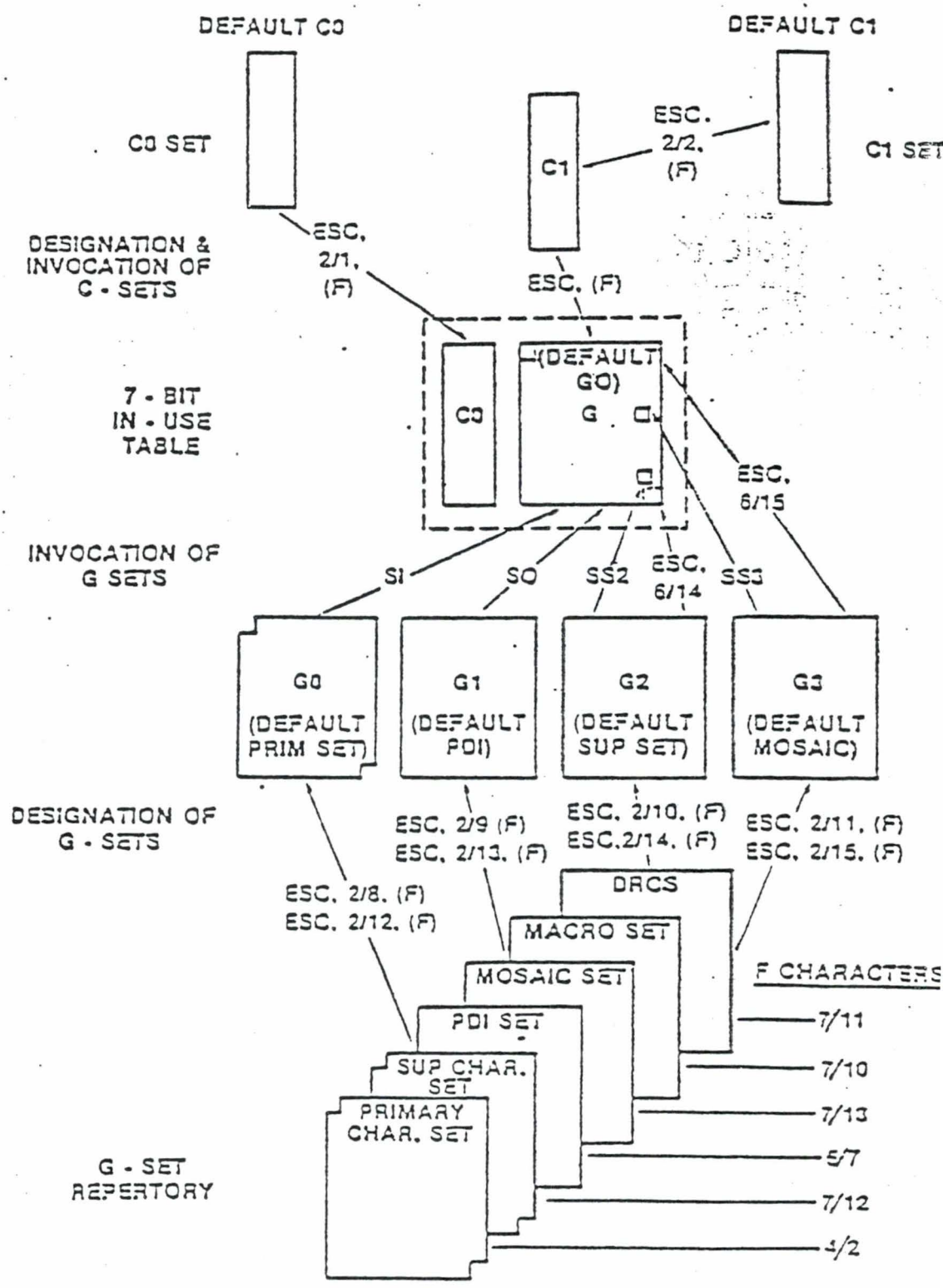


Figure 4

Code Extension in a 7-Bit Environment

A single additional control set, the C1 set, and 4 active G-sets, the G0, G1, G2, and G3 sets, are defined. The contents of the C1 set are described in Clause 6.2. The contents of the G0, G1, G2, and G3 sets can be dynamically selected from the larger repertory of G-sets described in Clauses 5 and 6 by using escape sequences. These sequences take the form ESC I F where I is the intermediate character and F is the final character. The intermediate character determines which set is to be changed (re-designated). Table 2 shows the assignment of I characters to the sets to be designated.

Table 2

Intermediate (I) Characters for Designation Sequences

Intermediate Character	Set to be Designated
2/1	C0
2/2	C1
2/8 or 2/12 *	G0
2/9 or 2/13 *	G1
2/10 or 2/14*	G2
2/11 or 2/15 *	G3

**There are two I characters that will result in the redesignation of each of four G sets. Only the first in each pair is defined for the designation now used. The remaining I character is reserved by the Registration Authority to enlarge the number of sets that may be registered.*

The final character determines which set from the larger repertory is to be selected. Table 3 shows the F character name assigned to each G-set. These are to be used in conjunction with the I characters 2/8, 2/9, 2/10, and 2/11. The F character for the Primary Character Set, for example, is 4/2. The 3 character escape sequence ESC 2/8 4/2, therefore, designates the Primary Character Set as the current G0 set.

The designation and invocation sequences for the C0 and C1 sets are ESC 2/1 F1 and ESC 2/2 F2, respectively, where F1 and F2 are to be assigned by the Registration Authority.

Table 3

Final (F) Characters for Designation Sequences of G-Sets

Final Character	G-Set
4/2	Primary Character Set
7/12	Supplementary Character Set
5/7	PDI Set
7/13	Mosaic Set
7/10	Macro Set
7/11	DRCS Set

The SI character is used to invoke the current G0 set into the in-use table where it remains until further control action is taken (i.e., it is invoked in a locking manner). The SO character is used to invoke the current G1 set into the in-use table in a locking manner. The sequence ESC 6/14 is used to invoke the G2 set into the in-use table in a locking manner. The sequence ESC 6/15 is used to invoke the G3 set into the in-use table in a locking manner. The single-shift characters, SS2 and SS3, are used to invoke, in a non-locking manner, the G2 or G3 set, respectively, into the in-use table. (The range of the single-shift characters extends only to the next character received, that is, the in-use table automatically reverts to its former state after the character immediately following the single-shift is interpreted.) Note that the PDI set can be single-shifted into the in-use table only in those cases where the PDI command is not to be followed by an associated numeric operand.

ESC 6/11, ESC 6/12, and ESC 6/13 are locking shifts intended primarily for use in 8-bits. However, their occurrence in 7-bits shall have the same meaning as SHIFT OUT (SO), ESC 6/14, and ESC 6/15, respectively, and may be used to preserve the sense of GL versus GR in case the 7-bit interchange is transformed to 8-bits.

The C1 set (in 7-bit environment) is never invoked into the in-use table in a locking manner. Rather, single characters from the C1 set are accessed via two character escape sequences. These sequences take the form, ESC Fe, where Fe represents the desired character from the C1 set. This character, by definition, must have a bit combination corresponding to column 4 or 5 of the 7-bit in-use table. As with the single-shift characters, the in-use table is not changed by these two character escape sequences. The in-use table automatically reverts to its former state after the C1 command is executed. (Note that although the C1 controls all consist of single characters, some commands may initiate multiple byte operations.)

If any of the G-sets are re-designated via an escape sequence while in the in-use table, the new code interpretations are simultaneously invoked, that is, a locking shift is not required for the change to take effect.

Upon initialization, the Primary Character Set (see Clause 5.1) is designated as the G0 set and the G0 set is invoked into the in-use table, by default. The PDI Set (see Clause 5.3) is designated as the G1 set, the Supplementary Character Set (see Clause 5.2) is designated as the G2 set, and the Mosaic Set (see Clause 5.4) is designated as the G3 set, all by default.

4.3.3 Code Extension in an 8-Bit Environment. When operating in an 8-bit environment, the 256 code positions available can also be extended to a much larger address space using similar code extension procedures as for the 7-bit environment. A 256 code position in-use table is defined, as shown in Figure 5. Again, each incoming bit combination is either decoded according to the contents of this table or is used to change the contents of this table. The table itself is organized into sixteen columns of sixteen rows, with bits 1 through 4 defining the row number and bits 5 through 8 defining the column number. The in-use table contains the C0 set in columns 0 and 1 and the C1 set in columns 8 and 9. Columns 2 through 7, which by convention will be called the GL (G-left) area of the in-use table, can accommodate any of the four invoked G-sets (G0, G1, G2, G3). Columns 10 through 15, which will be called the GR (G-right) area, can accommodate the G1, G2, or G3 sets. The manner in which this is accomplished is graphically depicted in Figure 6.

The SI character is used to invoke, in a locking manner, the G0 set into GL. Note that G0 cannot be invoked into GR. The SO character is used to invoke, in a locking manner, the G1 set into GL. The escape sequence ESC 6/11 is used to invoke, in a locking manner, the G1 set into GR. The escape sequences ESC 6/14 and ESC 6/12 are used to invoke, in a locking manner, the G2 set into GL and GR, respectively. Note also that the G2 set can be invoked into GL in a non-locking manner using the SS2 character, as described for use in a 7-bit environment. The escape sequences ESC 6/15 and ESC 6/13 are used to invoke, in a locking manner, the G3 set into GL and GR, respectively. Again, the G3 set can be invoked in a non-locking manner into GL using the SS3 character.

4.4 SPACE and DELETE. SPACE is an empty character field subject to the same attributes as alphanumeric characters. The coding is 2/0 in the 7- and 8-bit in-use table when a 94 code position G-set is invoked.

DELETE has been used primarily to erase or obliterate erroneous or unwanted characters in punched tape. The coding of DELETE is 7/15 in the 7-bit and 8-bit in-use table when a 94 code position G-set is invoked. DELETE is treated as a null operation in this Standard.

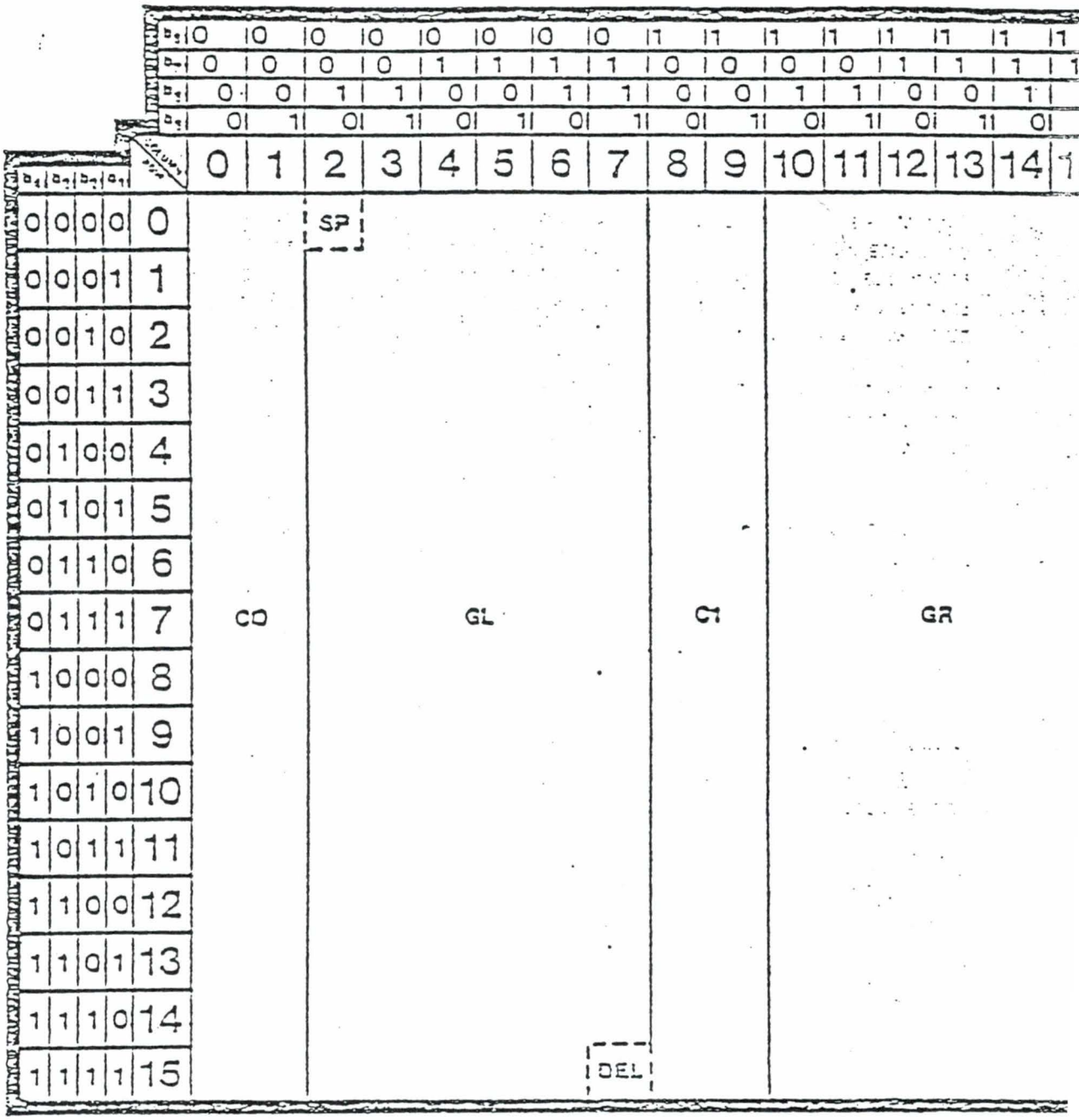


Figure 5

8-Bit In-Use Table

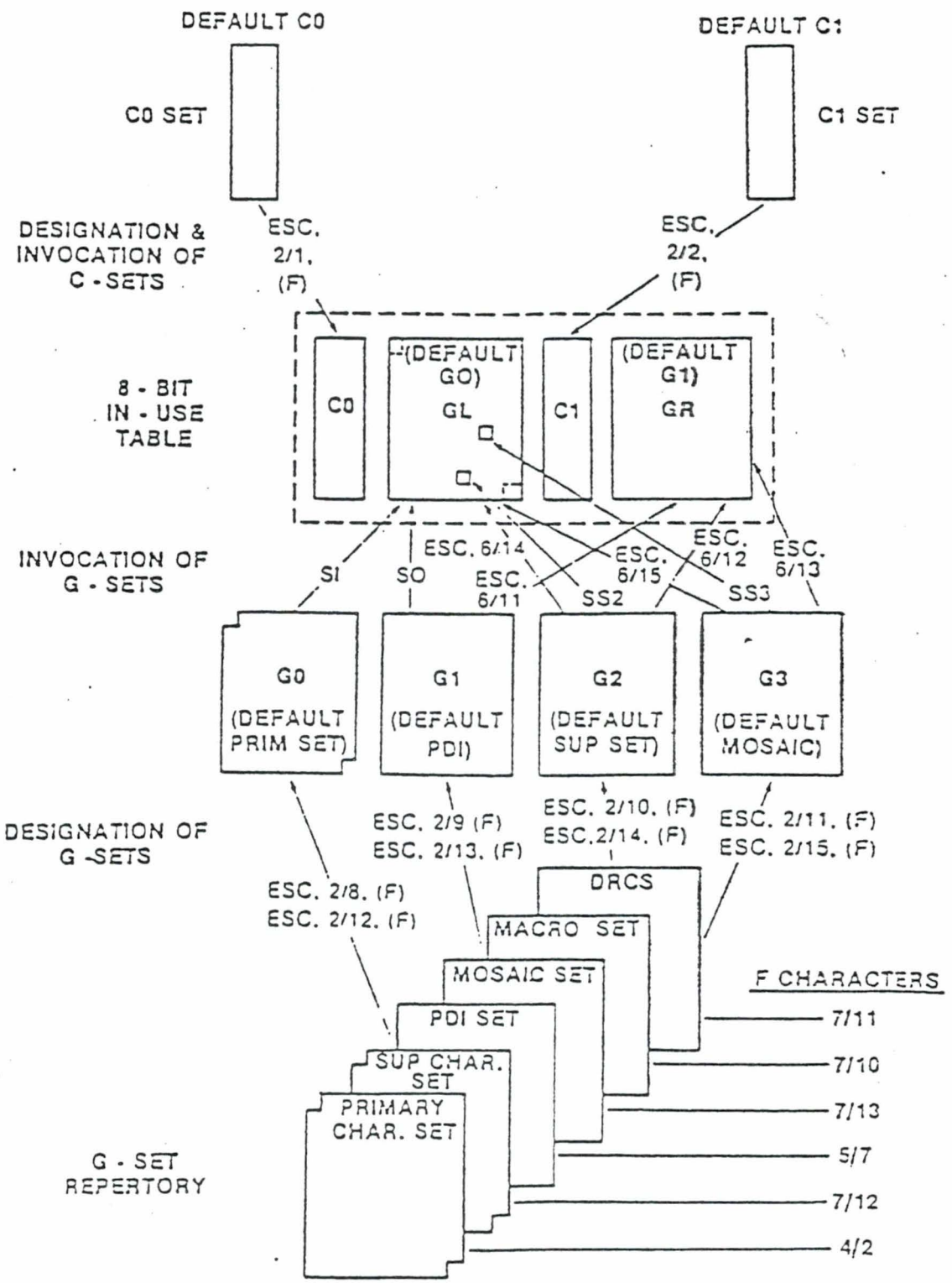


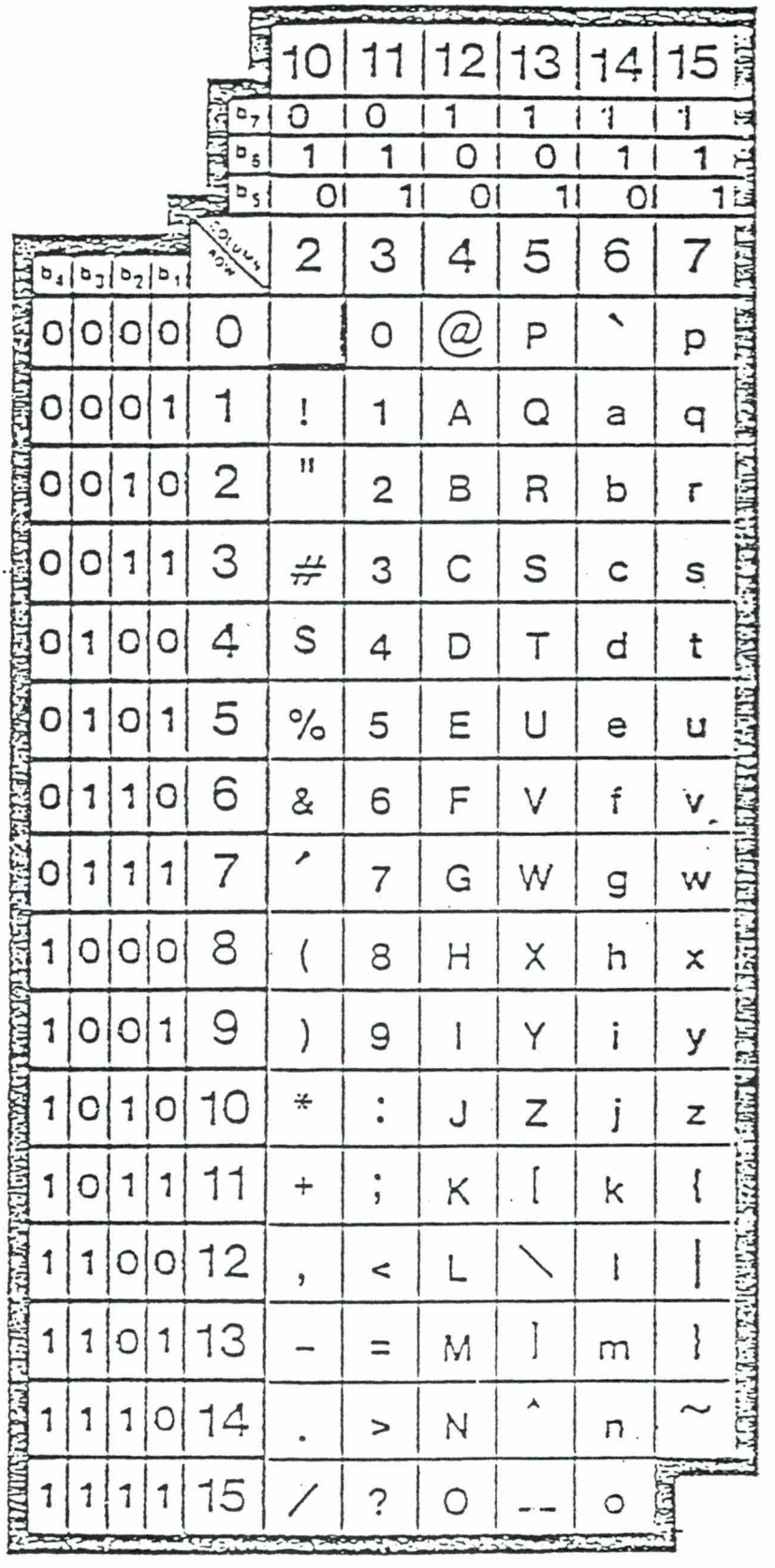
Figure 6

Code Extension in an 8-Bit Environment

5. Coding of G-Sets

5.1 Primary Character Set. Figure 7 shows the character assignments for the Primary Character Set. The particular patterns (font) chosen for the characters are not defined and are constrained only by the specified character field at each size for a given display resolution. Character legibility is not guaranteed at all sizes, in all colors and at all display resolutions. All characters of the Primary set are spacing.

The F character designation of the Primary Character Set, which is used for code extension purposes as defined in Clause 4.3, is 4/2.



b_4	b_3	b_2	b_1	ROW \ COLUMN	2	3	4	5	6	7
				b_7	0	0	1	1	1	1
				b_6	1	1	0	0	1	1
				b_5	0	1	0	1	0	1
0	0	0	0	0		0	@	P	`	p
0	0	0	1	1	!	1	A	Q	a	q
0	0	1	0	2	"	2	B	R	b	r
0	0	1	1	3	#	3	C	S	c	s
0	1	0	0	4	S	4	D	T	d	t
0	1	0	1	5	%	5	E	U	e	u
0	1	1	0	6	&	6	F	V	f	v
0	1	1	1	7	'	7	G	W	g	w
1	0	0	0	8	(	8	H	X	h	x
1	0	0	1	9	)	9	I	Y	i	y
1	0	1	0	10	*	:	J	Z	j	z
1	0	1	1	11	+	;	K	[	k	{
1	1	0	0	12	,	<	L	\	l	|
1	1	0	1	13	-	=	M	]	m	}
1	1	1	0	14	.	>	N	^	n	~
1	1	1	1	15	/	?	O	--	o	

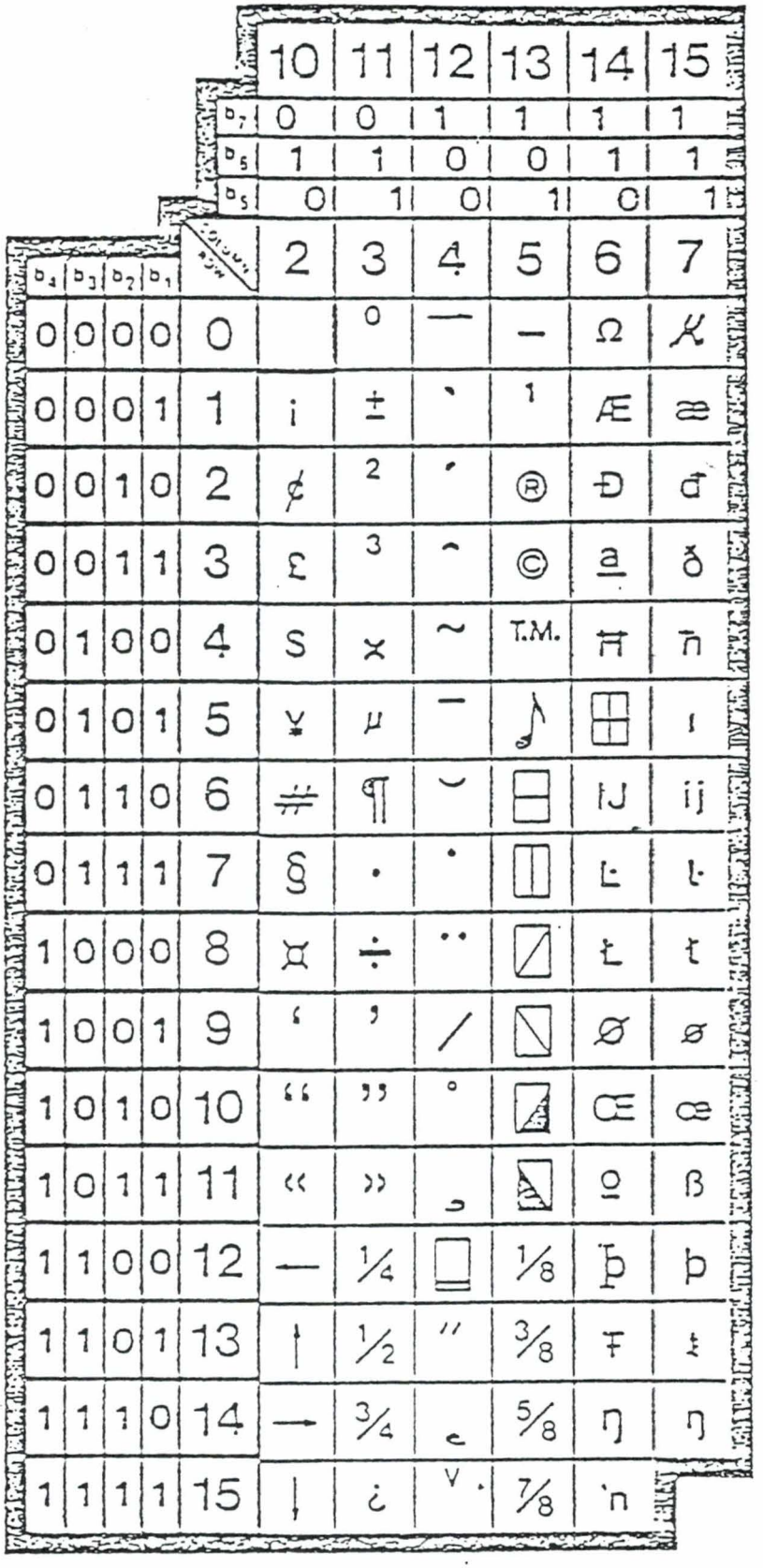
Figure 7

Primary Character Set

5.2 Supplementary Character Set. The Supplementary Character Set of accents, diacritical marks, and special characters for Latin-based alphabets is illustrated in Figure 8. This table is based on CCITT Recommendation S.100 and includes some additional characters proposed by CCIR, ISO, and other organizations. Those additional characters which are not yet included in a CCITT recommendation are in code positions 4/0, 4/9, 4/12, 5/0 to 5/15, and 6/5. The particular patterns (font) chosen for the characters are not defined and are constrained only by the specified character field at each size for a given display resolution. The 16 accent and symbol characters in Column 4 of the table are treated differently from all of the other characters in that they are non-spacing. That is, when one of these characters is received, the cursor is not automatically advanced as it would be normally, as described in Clause 5.3.2.3.

Coding for an accented character is obtained by composition of a non-spacing accent from the Supplementary set together with the letter from the Primary set. In typical usage a composite character would require three bytes to encode. For example, in a 7-bit environment with the Primary set designated in its default position as G0 and invoked into the in-use table and the Supplementary set designated as G2, the coding for ë (e with diaeresis) would be as follows. An SS2 (position 1/9 in the C0 set) would start the sequence invoking a single character from the code table G2. The diaeresis mark ¨ would then be specified, followed by the primary character. In such a manner the letter ë would be coded SS2 ¨ e, that is, three characters from code table positions 1/9, 4/8, 6/5. In an 8-bit environment, the coding may be the same as in the 7-bit environment or, if the Primary set is invoked into GL and the Supplementary set is invoked into GR, then the letter ë would be coded ¨ e, that is, the two characters from the code table 12/8, 6/5.

The F character designation of the Supplementary Character Set, which is used for code extension purposes as defined in Clause 4.3, is 7/12.



Note: Column 4 (12) is non-spacing.

Figure 8

Supplementary Character Set

5.3 Picture Description Instruction (PDI) Set

5.3.1 General

5.3.1.1 The Picture Description Instruction (PDI) set, shown in Figure 9, comprises six geometric graphic primitives (POINT, LINE, ARC, RECTANGLE, POLYGON, and INCREMENTAL), each of which has four forms; eight control codes (RESET, DOMAIN, TEXT, TEXTURE, SET COLOR, WAIT, SELECT COLOR, and BLINK); and 64 character positions for numeric data (corresponding to a 6-bit data field in each information byte.) The PDI set can be fundamentally differentiated from the alphanumeric character sets in that it does not consist of pre-defined patterns, one per character, but executable drawing functions that produce an image not necessarily restricted to a single character field.

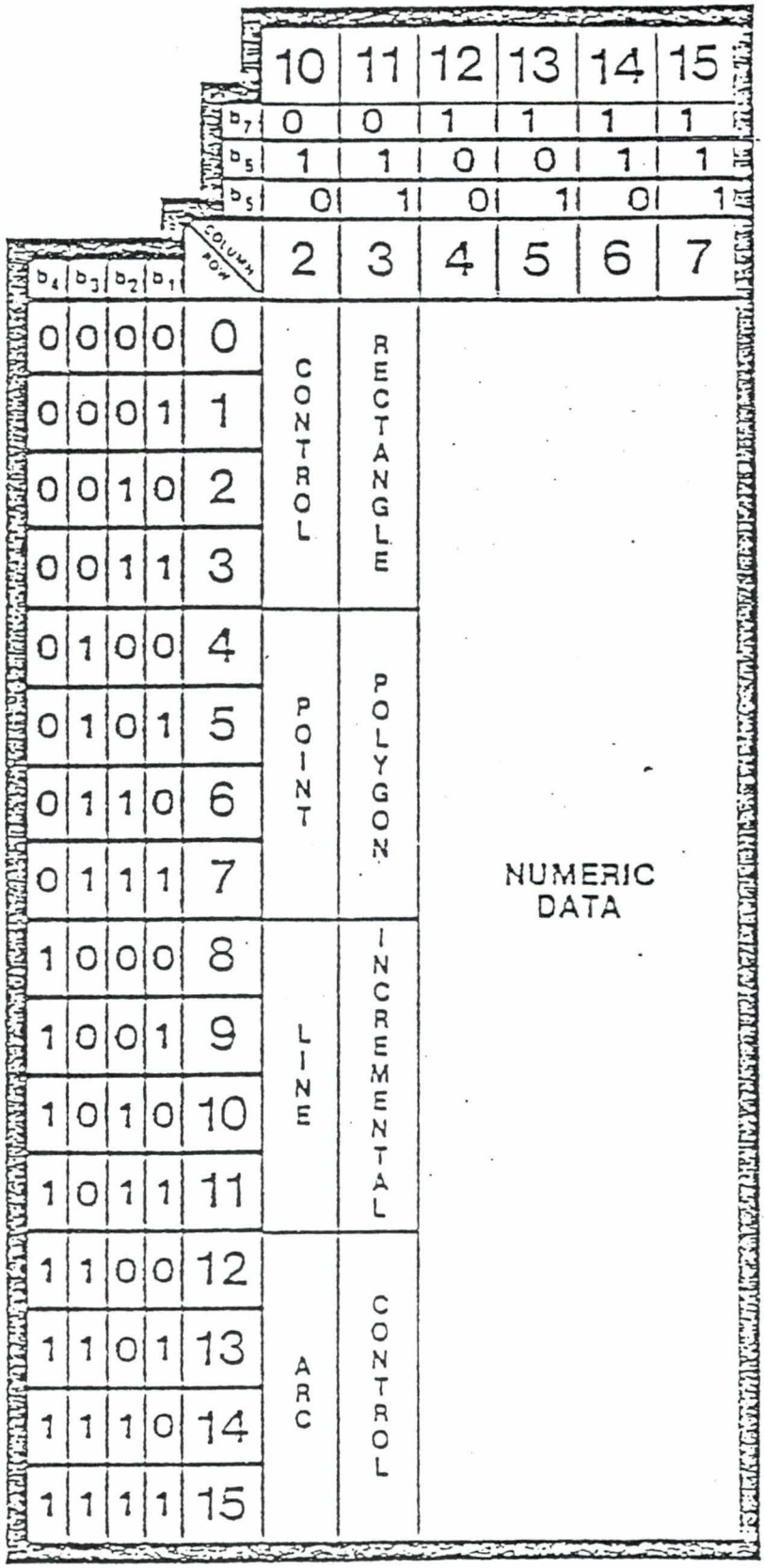


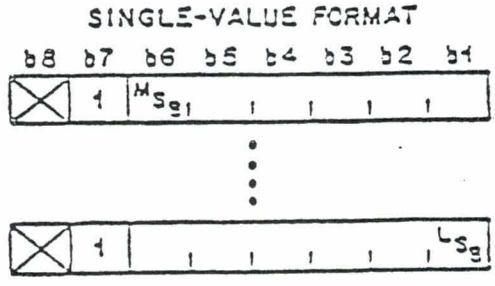
Figure 9

General PDI Set

However, the invocation of a macro either from the in-use table or by single shift will not by itself terminate the PDI - the PDI may be continued in the operand data contained in the macro.

A PDI is composed of an opcode, which must be either one of the four forms of the six graphic primitives or one of the eight control codes, followed by zero, one or more operands, each of which consists of one or more bytes of numeric data. The former can always be distinguished from the latter by examining bit 7. If b7 is set to 0, an opcode is indicated. If b7 is set to 1, numeric data (i.e., an operand) is indicated. A PDI sequence is terminated by an opcode introducing the next PDI sequence or by any other presentation layer code not from the numeric data section of the same PDI set, ~~with the exception of the invocation of a macro (either from the in-use table or by single shift) that contains only operand data.~~ There are four types of operands: fixed format, string, single-value, and multi-value.

The fixed format operands consist of one or more bytes of numeric data whose length and interpretation depends on the opcode with which they are used. The string operands are of indeterminate length, that is, they consist of any number of bytes of numeric data. Their interpretation also depends on the opcode with which they are used, but in all cases they are decoded left to right, i.e., b8 to b1. The single-value operands consist of one, two, three, or four bytes of numeric data, as determined by the DOMAIN command described in Clause 5.3.2.2. They are interpreted as unsigned integers (ordinal numbers) composed of the sequence of concatenated bits taken consecutively (high order bit or b8 to low order bit or b1) from the numeric data bytes as shown in Figure 10.



MSB - MOST SIGNIFICANT BIT OF OPERAND

LSB - LEAST SIGNIFICANT BIT OF OPERAND

Most Significant Byte transmitted first

Figure 10

Single-Value Format

The multi-value operands consist of one, two, three, four, five, six, seven, or eight bytes of numeric data, as determined by the DOMAIN command. These operands are used to specify coordinate information, when used in conjunction with the graphic primitives, or color information, when used in conjunction with the SET COLOR command.

The coordinate specifications are based on a unit Cartesian numbering scheme with positions being specified as fractions of this range from 0 (inclusive) to 1 (non-inclusive).

The coordinate data defined by the PDI operands can either be interpreted as the absolute coordinates within the unit screen of a logical drawing point or as displacements from the previous drawing point, depending on the context defined by the particular opcode. This drawing point is then used in the execution of the geometric primitives, described in Clause 5.3.3.

The representation of the coordinate data within the multi-value operand is shown in Figure 11.

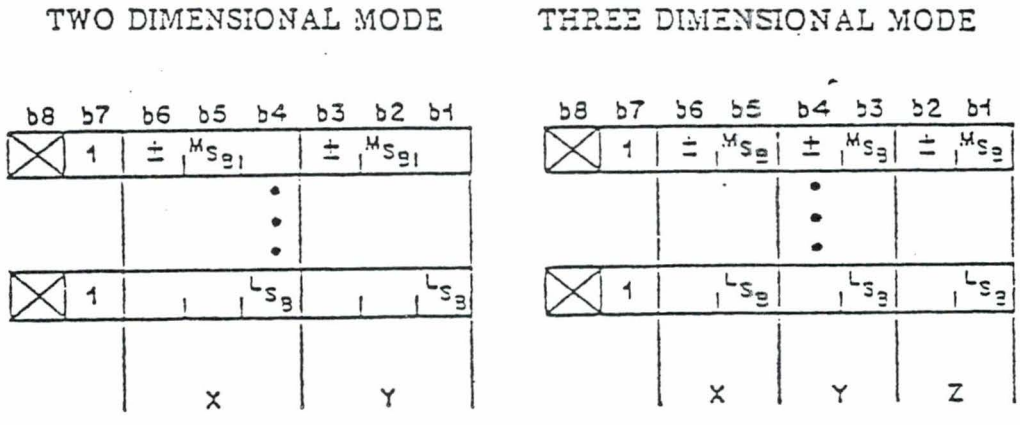


Figure 11

Multi-Value Format

The precision to which the position of the cursor and drawing point must be maintained is implementation dependent and should be consistent with the physical resolution.

All coordinate operands are interpreted as signed, two's complement numbers, i.e., binary decimals where the MSB represents the digit just to the right of the decimal point. If a coordinate specification or a drawing operation would cause the drawing point or any portion of the resulting drawing to be outside the unit screen, then the PDI is considered to be in error. ~~This~~ PDI may either be rejected (i.e., executed as a null ~~or no-op~~) or executed and clipped within the unit screen, ~~depending on the implementation.~~ operation

When the multi-value operand is used along with the SET COLOR control, described in Clause 5.3.2.5, it specifies an unsigned color value in the RGB (red-green-blue) color system. The representation of the color data within the multi-value operand is shown in Figure 12.

The handling of this error condition is implementation dependen
For example, this

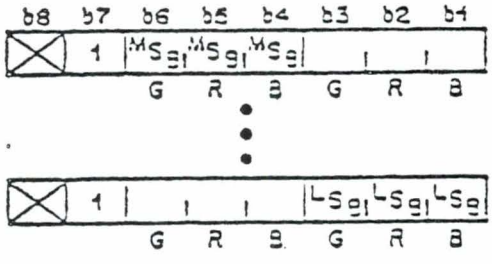


Figure 12
Color-Value

Note that each byte contains two three-tuples. Each three-tuple contains one bit for each of the three primary colors. These are specified in the order green, red, blue. A complete color value for each primary consists of the concatenated bits, taken one from each three-tuple, starting with the indicated MSB and proceeding, left to right, to the indicated LSB. The color value thereby obtained represents a binary fraction in which the MSB acts as the digit just to the right of the decimal point.

Table 4 shows the types of operands used by each of the opcodes.

Table 4

Operand Types

Opcode	Operand
RESET	fixed
DOMAIN	fixed/multi-value
TEXT	fixed/multi-value
TEXTURE	fixed/multi-value
SET COLOR	multi-value
WAIT	fixed
SELECT COLOR	single-value
BLINK	single-value/fixed
POINT	multi-value
LINE	multi-value
ARC	multi-value
RECTANGLE	multi-value
POLYGON	multi-value
FIELD	multi-value
INCREMENTAL POINT	fixed/string
INCREMENTAL LINE	multi-value/string
INCREMENTAL POLYGON (FILLED)	multi-value/string

5.3.1.2 The functions of the opcodes are summarized as follows:

(a) POINT sets the drawing point to any position in the display space and optionally displays a dot;

(b) LINE draws a line based on its endpoints;

(c) ARC draws a circular arc based on the endpoints of the arc and a point on the arc. The endpoints of the arc may optionally be joined by a chord and the area so defined filled in. If more points are given, they define a higher level

arc, a curvilinear line defined by a spline function. A circle is described as an arc whose endpoints coincide and whose intermediate point (with the endpoints) define the diameter;

(d) RECTANGLE draws a rectangular outline or fills in an area of specified length and width;

(e) POLYGON draws a polygonal outline or fills in the circumscribed area based on a series of defined vertices;

(f) INCREMENTAL draws a point, line, or polygon in an incremental manner;

(g) CONTROL provides control over the modes of the drawing commands. One of its major functions is to set up a value or color of an object.

Figure 13 shows the detailed layout of the PDI Set with each form of the geometric primitives and control codes identified.

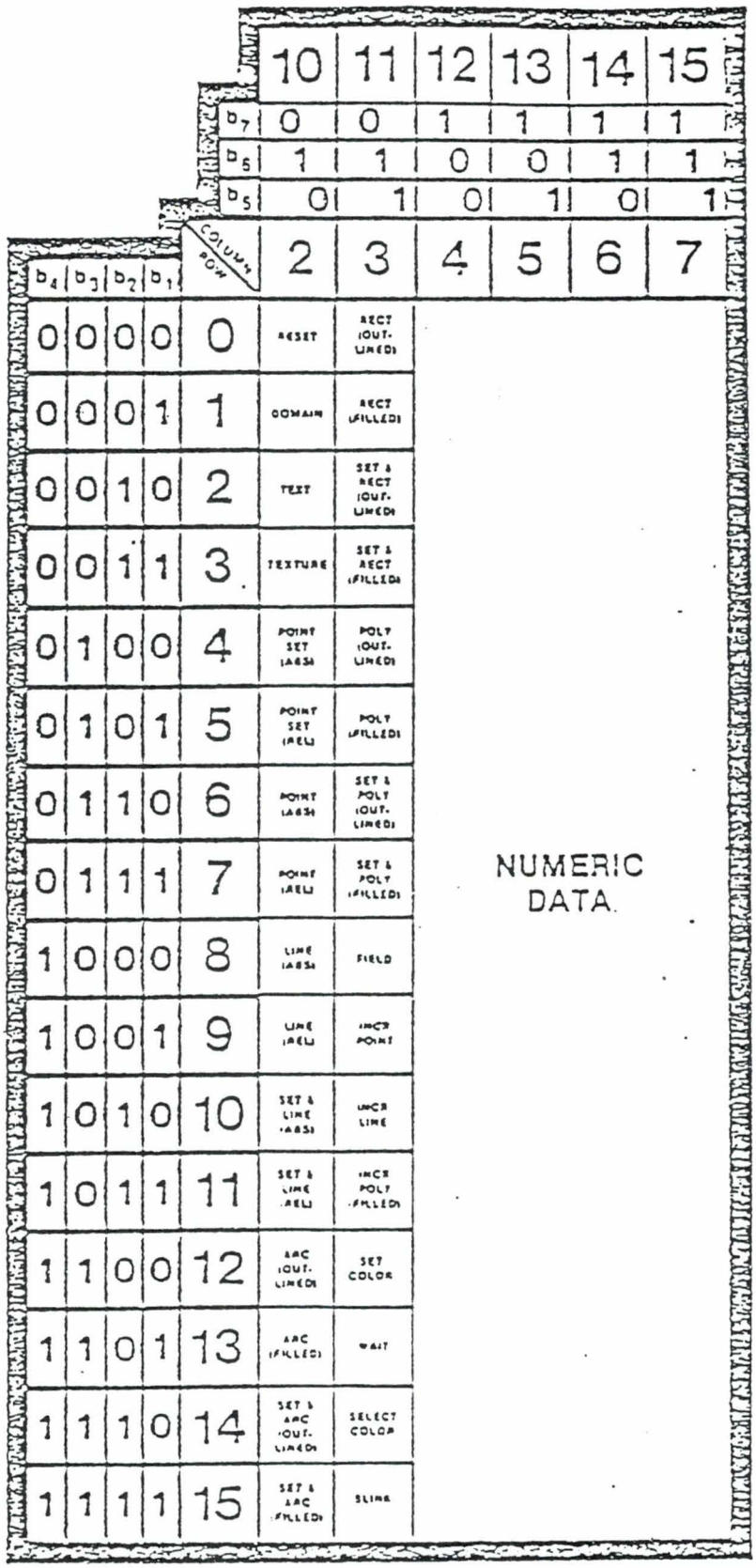


Figure 13

PDI Set

5.3.2 Attribute Control Functions

5.3.2.1 General. The opcodes described in Clauses 5.3.2.2 to 5.3.2.9 control the attributes and display parameters.

5.3.2.2 DOMAIN

5.3.2.2.1 Command Format. The DOMAIN command is used to control operand parameters for subsequently received coordinate data, Red-Green-Blue (R-G-B) color specifications, and color map addresses as well as the size of the logical pel to be defined. (See Figure 14.) Once set, these parameters do not change until acted upon by either the RESET command, another DOMAIN command, or the NSR control code described in Clause 6.1.5.15. The DOMAIN opcode takes a one byte, fixed format operand, followed by a multi-value operand, whose interpretations are shown below.

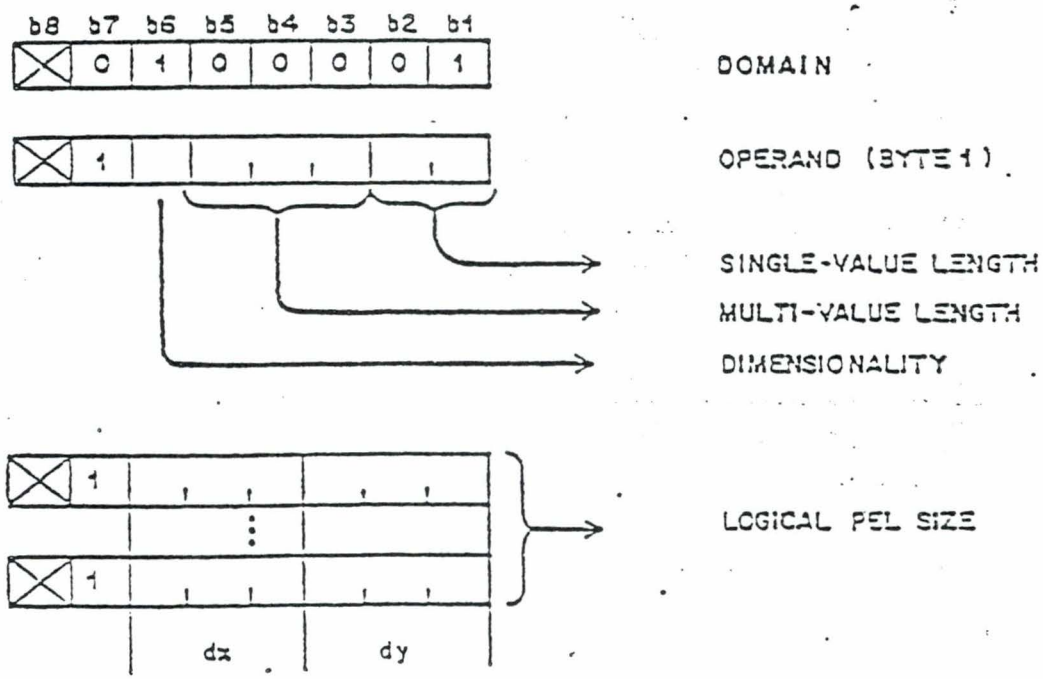


Figure 14

5.3.2.2.2 Single-Value Operand ~~Link~~ Length Bits b1 and b2 of byte 1 determine the length, that is, the number of bytes to be used in single-value operands according to Table 5. The default length is one byte.

Table 5

b2	b1	Single-Value Operand Length (Bytes)
0	0	1 (default value)
0	1	2
1	0	3
1	1	4

5.3.2.2.3 Multi-Value Operand ~~Link~~ Length Bits b3, b4, and b5 of byte 1 determine the length, that is, the number of bytes to be used in multi-value operands according to Table 6. The default length is three bytes.

Table 6

b5	b4	b3	Multi-Value Operand Length (Bytes)
0	0	0	1
0	0	1	2
0	1	0	3 (default value)
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

5.3.2.2.4 Dimensionality. Bit 6 of byte 1 determines the dimensionality of the coordinate specification. A 0 indicates two-dimensional (X,Y) mode, which is the default. A 1 indicates three-dimensional (X,Y,Z) mode. If three-dimensional coordinates are received, the Z coordinate is to be ignored, thereby projecting the image into the two-dimensional (X,Y) plane. The full definition of three-dimensional mode is reserved for future standardization.

5.3.2.2.5 Operand ~~Disk~~ Length. If an operand following an opcode is shorter than the length previously specified by the DOMAIN command (or the implicit length in the fixed format case), then the remainder of the operand is padded with zeros, unless otherwise indicated in the definition of the command. If an operand following an opcode is longer than the length previously specified by the DOMAIN command (or the implicit length), it is taken as an indication to repeat the execution of the opcode with the subsequent numeric data taken as new operands, unless otherwise indicated in this Standard.

5.3.2.2.6 Logical Pel. The coordinate data following byte 1 of the operand is interpreted to be the width (dx) and height (dy) of the logical pel, which is a rectangle whose orientation is fixed with respect to the Cartesian coordinate system. This multi-value operand specifies the logical pel size to be used with the POINT, LINE, ARC, RECTANGLE, POLYGON, and INCREMENTAL PDFs but not for the display of alphanumeric characters. This is accomplished by defining the drawing operations to affect all of those pixels that lie under any portion of the logical pel as it is mapped to the display screen. The logical pel, therefore, will always map to at least one and possibly many display pixels. Note that if the width and height of the logical pel are both reduced to 0, the logical pel reduces to the dimensionless drawing point. The default logical pel size is dx = +0, dy = +0.

A drawing primitive is defined by an algorithm that describes as closely as possible a precise geometric path. For example, a LINE is a locus of points following a straight line algorithm between two specified coordinates. The physical picture elements (pixels) through which the infinitely small locus point passes would be drawn. The logical pel specification allows the locus point to take on specific dimensions, thereby acting as a larger "brush" that turns on additional pixels as it traverses its geometric path and generates the effect of line width. See Figure 15, which is illustrative of the application of logical pel to an arc.

The geometric alignment of the drawing point within the logical pel is:

(a) Lower left corner if both dx and dy are positive;

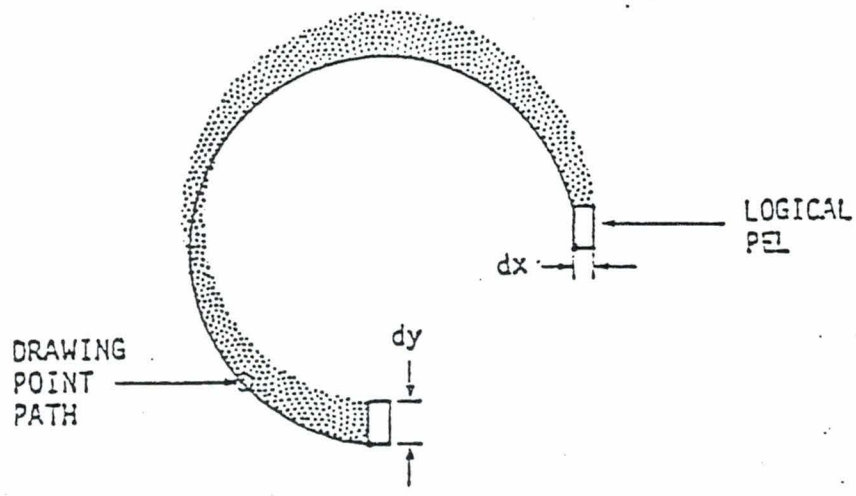
(b) Lower right corner if dx is negative and dy is positive;

(c) Upper left corner if dx is positive and dy is negative; and

(d) Upper right corner if both dx and dy are negative.

Note that the new length of the multi-value operands, as set in byte 1, applies to the multi-value logical pel size operand of that DOMAIN command.

If additional numeric data bytes follow the logical pel size data byte(s), they are ignored.



Effect of logical pel size on
stroke width in ARC command

Figure 15

Application of Logical Pel to an Arc

5.3.2.3 TEXT

5.3.2.3.1 Command Format. This opcode is used to modify parameters that describe the manner in which subsequent alphanumeric characters, mosaic characters, and DRCS are presented. The TEXT opcode takes a two byte, fixed format operand, followed by a multi-value operand whose interpretations are shown below. (See Figure 16.)

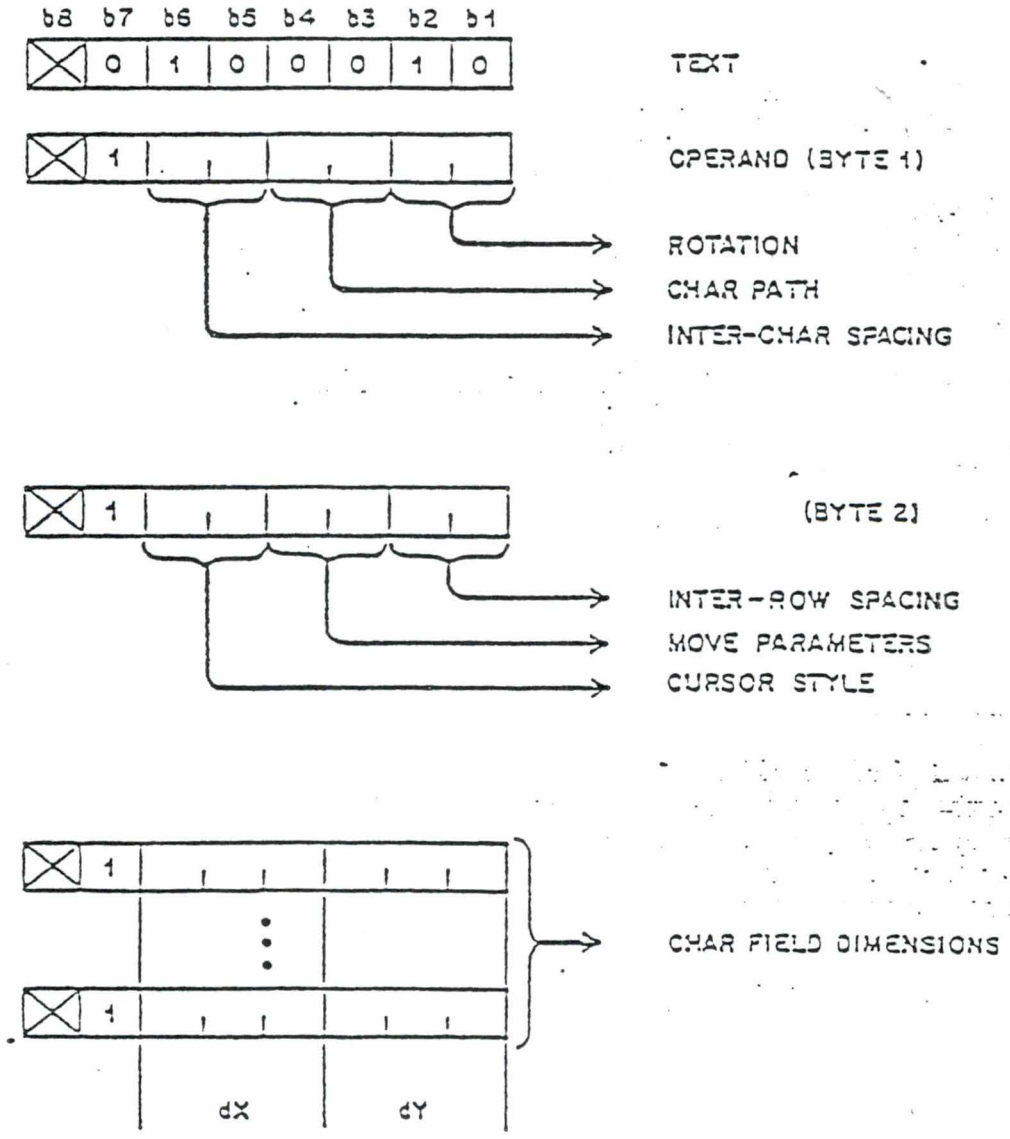

b8 b7 b6 b5 b4 b3 b2 b1
0 1 0 0 0 1 0
TEXT
1
OPERAND (BYTE 1)
ROTATION
CHAR PATH
INTER-CHAR SPACING
1
(BYTE 2)
INTER-ROW SPACING
MOVE PARAMETERS
CURSOR STYLE
1
1
CHAR FIELD DIMENSIONS
dX
dY

Figure 16

5.3.2.3.2 Character Rotation. Bits b1 and b2 of byte 1 are used to specify character rotation as shown in Table 7.

Table 7

b2	b1	Rotation
0	0	0° (default value)
0	1	90°
1	0	180°
1	1	270°

Rotation causes the character field and the cursor to rotate counter-clockwise about the character field origin. This rotation is measured relative to horizontal within the unit screen and is independent of the character path (charpath). The character field origin is the lower left corner of the character field at the default 0° rotation (see Figure 17). All images (e.g., alphonumeric characters including diacritical marks, underlines, DRCS, mosaics and separated mosaics) a character field are affected by rotation 0° so that the relative positio of the images within the character field is unchanged.

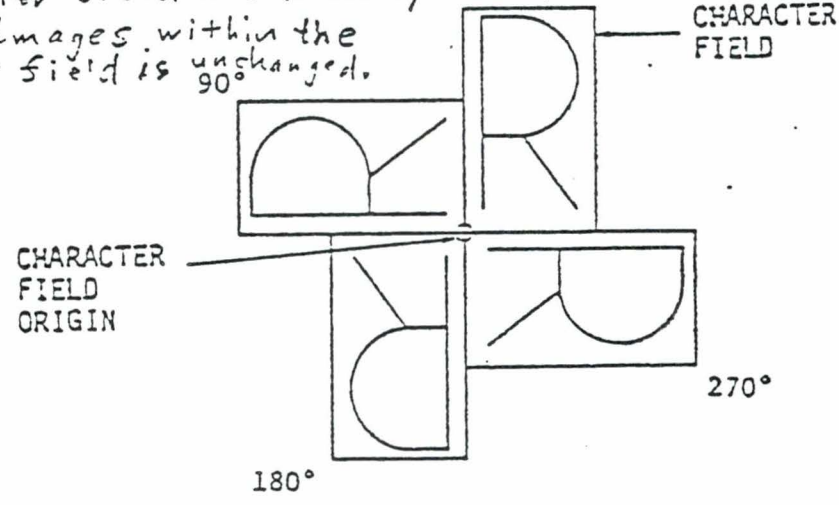


Figure 17

Character Rotation

5.3.2.3.3 Character Path Movement. Bits b3 and b4 of byte 1 determine the direction of the character path, that is, the direction in which the cursor is automatically advanced after a character is deposited. Table 8 describes the four possible charpaths. The charpath is defined relative to horizontal within the unit screen and is independent of the character rotation. The default charpath is right.

Table 8

b4	b3	Character Path Movement
0	0	Right (default).
0	1	Left.
1	0	Up.
1	1	Down.

5.3.2.3.4 Inter-Character Spacing. Bits b5 and b6 of byte 1 are used to determine the distance the cursor is moved (in multiples of the character field dimension lying parallel to the character path) after a character is displayed or after a SPACE or APB (backspace) or APF (horizontal tab) character is received. This is known as the inter-character spacing and is as defined in Table 9.

Table 9

b6	b5	Inter-character Spacing
0	0	1 (default value)
0	1	1.25
1	0	1.5
1	1	Proportional spacing

that are applied to the cursor movement

The three fixed inter-character spacings (1, 1.25, and 1.5) are interpreted as multiplicative functions of the dimension of the current character field that lies parallel to the charpath. In proportional spaced mode, the inter-character spacing is a variable that may be a function of the width of the actual pattern deposited as well as the current character size and font style. The proportional spacing algorithm is ~~not explicitly specified but is left to the design of the terminal manufacturer.~~ This means that the exact number of characters per line is not known in proportional spacing mode, but it ~~will~~ at least as many characters per line as would be allowed by the current character field dimensions. The default inter-character spacing is a fixed space of 1.

5.3.2.3.5 Inter-Row Spacing. Bits b1 and b2 of byte 2 determine the inter-row spacing of characters, which defines the relative location of the cursor when it is advanced to a new line in a direction perpendicular (-90°) to the charpath, either automatically as described below or by the APD (line feed) or APU (vertical tab) characters, as defined in Clause 6.1.5. Table 10 shows the alternative inter-row spacings, which are interpreted as multiplicative functions of the dimension of the character field that lies perpendicular to the charpath.

However, each character shall be completely contained within the area defined by the current character field (see Clause 5.3.2.3.8).

When using fixed or proportional inter-character spacing, if the character field origin is advanced along the charpath such that any part of the corresponding character field would exceed the edge of the unit screen or the active field (See Clause 5.3.3.6.2, which describes FIELD), an automatic APR (carriage return) and APD (line feed - see Clause 6.1.2.3) are executed.

Table 10

b2	b1	Inter-row Spacing
0	0	1 (default value)
0	1	1.25
1	0	1.5
1	1	2

~~If the cursor is advanced along the charpath past the edge of the unit screen or the active drawing area (see Clause 5.3.3.6.2, which describes FIELD), when using a fixed inter character spacing, an automatic carriage return and line feed are executed. When using the proportional inter character spacing, an automatic carriage return and line feed are executed only when the character to be deposited does not fit in the remaining space between the previous character pattern and the edge of the unit screen or active drawing area. The automatic carriage return and line feed should not occur until an attempt is actually made to display a character that does not fit.~~ An inter-row spacing of 1, in which the character field on the current row abuts the character field of the previous row, is the default.

5.3.2.3.6 Move Attributes. Bits b3 and b4 of byte 2 are used to define the relationship between movement of the cursor and movement of the graphics drawing point as shown in Table 11.

If an automatic APR APD is made extraneous by a subsequently received APR APD sequence, only one APR APD is executed and presented.

Table 11

b4	b3	Move Attribute
0	0	Move together (default).
0	1	Cursor leads.
1	0	Drawing point leads.
1	1	Move independently.

If the cursor and drawing point are set to move together (00), then whenever the cursor is moved (such as when characters are displayed) the graphics point is moved with it, maintaining its alignment relative to the cursor. Correspondingly, whenever the drawing point is moved (such as with a geometric drawing primitive) the cursor is also moved so as to maintain its alignment relative to the drawing point.

The execution of a TEXT command shall cause alignment of the drawing point is the 'move together' or 'cursor leads' move attribute is in effect after execution. The execution of a TEXT command shall have no effect on the position of the character field origin.

If the cursor is defined as leading (01), then every time the cursor is moved the drawing point will move along with it but not vice versa.

If the drawing point is set to lead (10) the cursor, then every time the drawing point is moved the cursor will move with it but not vice versa.

If the drawing point and the cursor are set to move independently (11), then movement of one will not affect the position of the other.

Movement of the drawing point should never cause the cursor to be located such that any part of the character field indicated by the cursor would fall outside the unit screen. Should such a situation arise, the cursor will be ~~moved~~ adjusted as close as possible to the drawing point without violating the above condition.

The alignment of the drawing point corresponds to the character field origin for the underscore cursor and block cursor, and the center of the character field for the crosshair cursor and custom cursor. (See Figure 18.)

5.3.2.3.7 Cursor Styles. Bits b5 and b6 of byte 2 are used to determine the display style of the cursor as in Table 12 and Figure 18.

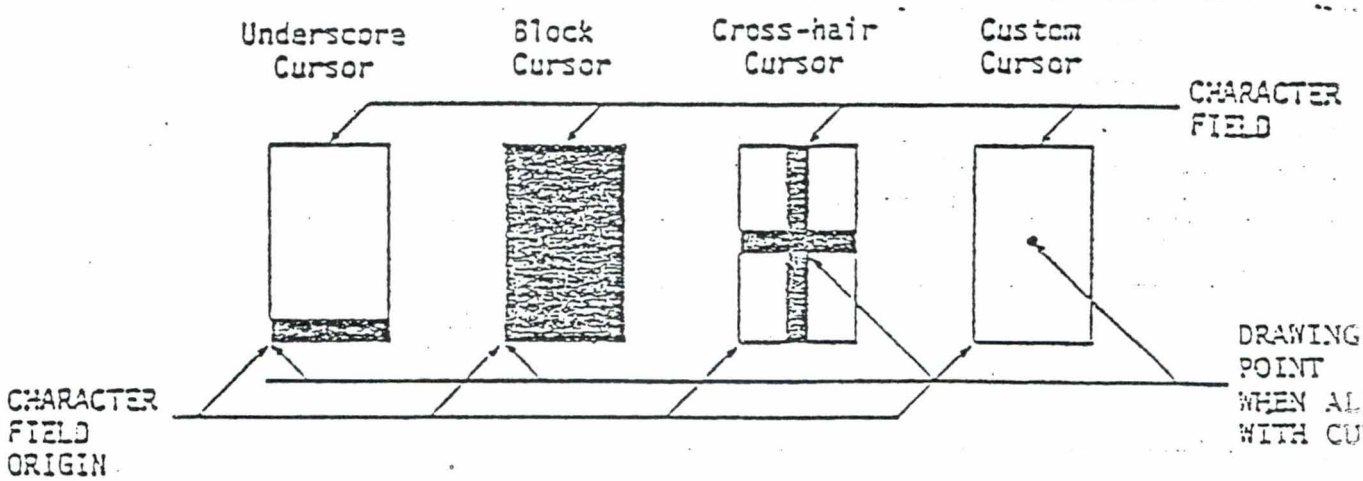


Figure 18

Cursor Styles

Subsequent relative cursor positioning operations shall be made with reference to the adjusted cursor posi

Table 12

b6	b5	Cursor Style
0	0	Underscore (default style).
0	1	Block.
1	0	Cross-hair.
1	1	Custom.

The cursor is located in the position in which the next character is to be deposited. The underscore cursor is a single line the width of the current character field at the bottom of the character field. The block cursor is a solid block whose size is the size of the current character field. The cross-hair cursor consists of a vertical line and a horizontal line that intersect at the center of the character field and whose height and width are equal to the height and width of the current character field. The definition of the shape of the custom cursor is ~~left to the terminal manufacturer.~~

5.3.2.3.8 Character Field Dimensions. The multi-value operand data following the first two fixed format operands gives the width (dx) and height (dy) of the character field.

If dx is negative, the character patterns are reflected about the vertical center axis of the character field. If dy is negative, the character patterns are reflected about the horizontal center axis of the character field.

If the character field dimensions are omitted from the operand, then the current character field dimensions remain unchanged.

The default dimensions of the character field are dx = approximately 0.025 (i.e., dx = approximately 0.00000110 binary) and dy = approximately 0.04 (i.e., dy = approximately 0.00001010 binary).

If additional numeric data follows the multi-value operand, the additional numeric data is ignored.

A presentation process shall not display text characters larger than the current character field dimensions.

The font and position of alphanumeric text characters within the character field is implementation dependent. Each such character shall be completely contained within the area defined by the current character field.

5.3.2.4 TEXTURE

5.3.2.4.1 Command Format. This opcode is used to set texture attributes that are applied to the subsequent drawing of lines, the highlighting of filled areas, and the patterns used to fill areas (see Figure 19).

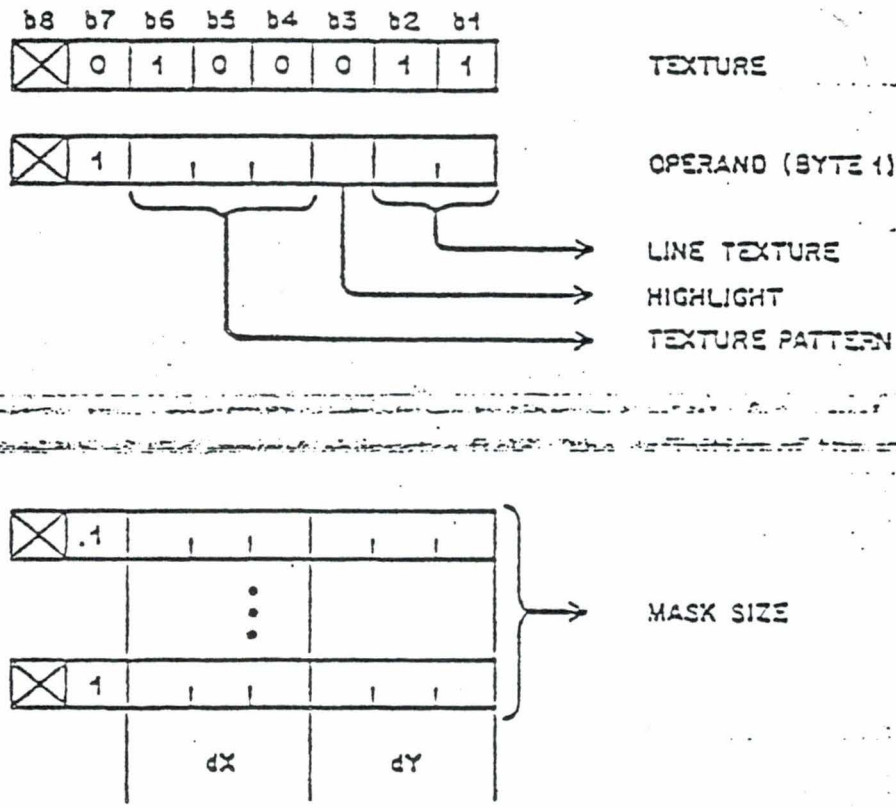


Figure 19
TEXTURE

5.3.2.4.2 Line Texture. Bits b1 and b2 of byte 1 are used to set the line texture attribute, which determines the style of lines and outlines (but not highlights) drawn with the LINE, ARC, RECTANGLE, POLYGON, and INCREMENTAL LINE PDIs (see Figure 20 and Table 13). The size of the dot is set equal to the size of the logical pel. For horizontal lines, the inter-dot spacing is the width of the logical pel, while for vertical lines it is the height of the logical pel. For horizontal lines, the height of the dash is equal to the height of the logical pel while the width (length) of the dash and the inter-dash spacing are equal to three times the width of the logical pel. For vertical lines, the width of the dash is equal to the width of the logical pel while the height (length) of the dash and the inter-dash spacing are equal to three times the height of the logical pel. The inter-dot-dash spacing is equivalent to the inter-dot spacing.

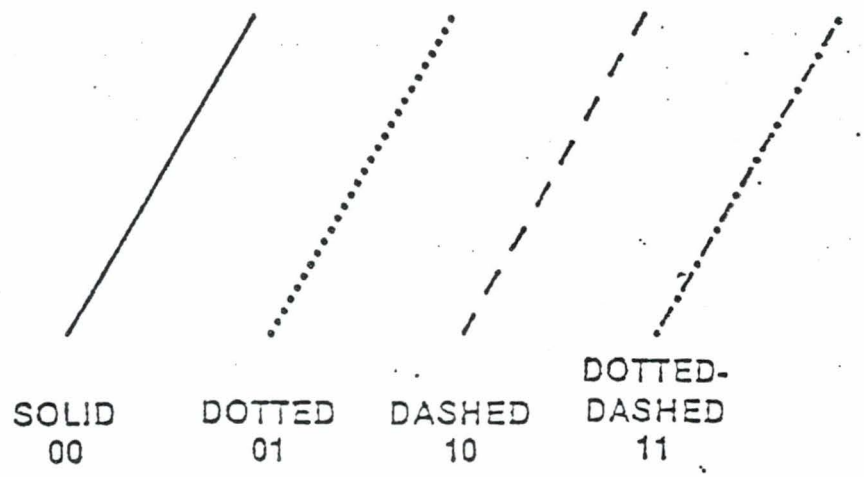


Figure 20

Line Textures

The algorithm for generating line textures is left to the equipment designer, and it should produce characteristics for arbitrary lines that yield a visually consistent effect with that specified for horizontal and vertical lines. All endpoints of lines and arcs and all vertices of rectangles and polygons must be plotted regardless of the line texture used. If the logical pel size is 0, solid lines will always be drawn.

Table 13

b2	b1	Line Texture
0	0	Solid (default).
0	1	Dotted.
1	0	Dashed.
1	1	Dotted-dashed.

5.3.2.4.3 Highlight. Bit b3 of byte 1 determines the highlight attribute. If bit 3 is equal to 1, then all filled rectangles, arcs, polygons, and incremental polygons are drawn in highlighted mode. In this mode, perimeters are drawn with solid line texture using the current logical pel size in nominal black in color modes 0 and 1, and in the background color in color mode 2. The default state of this attribute is no highlight (b3 = 0). (See Clause 5.3.2.6 for a description of the three color modes.)

5.3.2.4.4 Texture Pattern. Bits b4, b5, and b6 are used to select the texture pattern to be used in filling rectangles, arcs, polygons, and incremental polygons according to Table 14 and Figure 21.

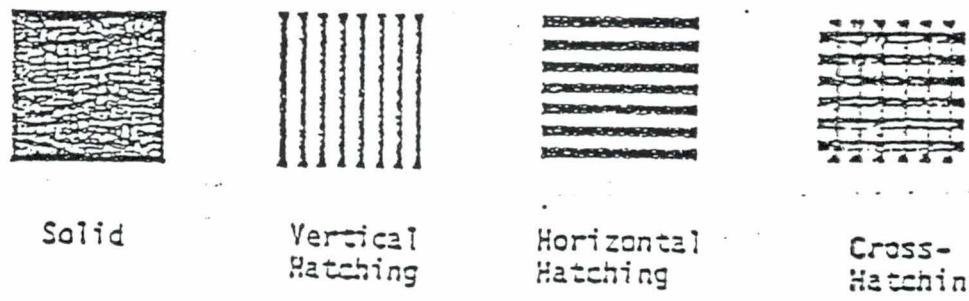


Figure 21

Texture Patterns

Table 14

b6	b5	b4	Texture Pattern
0	0	0	Solid (default pattern).
0	0	1	Vertical hatching.
0	1	0	Horizontal hatching.
0	1	1	Vertical and Horizontal Cross-hatching.
1	0	0	Mask A.
1	0	1	Mask B.
1	1	0	Mask C.
1	1	1	Mask D.

The width and spacing of hatching lines in the vertical hatching pattern are equal to the width of the logical pel. The height and spacing of hatching lines in the horizontal hatching pattern are equal to the height of the logical pel. Registration of the patterns must be maintained across figures. If logical pel size is 0, solid texture patterns will always be shown. The programmable texture masks A, B, C, and D are defined using the DEF TEXTURE command.

5.3.2.4.5 Mask Size. The block of coordinate data following the first byte of the operand specifies the mask size (dx, dy) to be used in the step-and-repeat process for masks A, B, C, and D. This process takes the selected texture mask, scales it to the specified mask size, logically covers the given object with contiguous copies of the mask, and then deposits the in-use color(s) in all pixels indicated by the mask pattern. This process takes as its initial reference the origin (0,0) point of the unit screen in order that registration of the pattern may be maintained across figures at any given mask size.

The default mask size is dx = approximately 0.025, dy = approximately 0.04 (the default character field size). The sign bits of dx and dy are used to reflect the mask pattern within the mask field in a manner similar to reflection of text character fields.

If the mask size operand is not present within the TEXTURE PDI, then the current mask size is not changed. If additional numeric data follows the mask size operand, the additional numeric data is ignored.

5.3.2.5 SET COLOR

5.3.2.5.1 The SET COLOR command is used to specify color values applied to all subsequent drawing commands and alphanumeric characters. Three different color modes can be selected, the choice of which dictates the precise interpretation of the two color control opcodes, SET COLOR and SELECT COLOR. The color mode can be set to 0, 1, or 2 via the SELECT COLOR PDI as described in Clause 5.3.2.6. Color mode 0 is designed to support those situations in which the in-use drawing color is directly specified as a color value. Colors ~~become~~ are implicitly defined in a color map in this mode. Color modes 1 and 2 are designed to make explicit use of a color map capability. That is, the in-use drawing color is specified as an ordinal number that is used as an address into a look-up table that provides the actual color value.

To illustrate the differences between the three color modes, consider the example of writing text. In color mode 0 the color is *set* directly and then applied *only* to the foreground pixels, i.e., only to the pixels that comprise the character pattern. In color mode 1 the color is *selected* from the color map, and again applied *only* to the foreground pixels. In color mode 2 *both* foreground and background colors are *selected* from the color map and then applied to the foreground and background pixels respectively.

The color map is used to convert, at display time, the color map address stored for each pixel in the display storage medium into an actual color value for that pixel. The number of bits (N) in the color map address (i.e., the number of bits per pixel in the display storage medium) is, by design, smaller than the number of bits (M) in the actual color value stored in the color map (i.e., the width of the color map). This provides, among other things, an increase in the total number of possible display colors (up to 2^M) without an increase in the size of the display storage medium, with the constraint that not more than 2^N colors can be displayed simultaneously.

Completely defining a color in color mode 1 and 2 takes two steps. The color values stored in the color map must be specified and the color map address (i.e., the ordinal number) to be associated with the in-use drawing color must be specified. In color modes 1 and 2, the SET COLOR control performs the former function and the SELECT COLOR control performs the latter function. Note that the color map applies to the entire display. A change in the color map will immediately be reflected in the color of all pixels whose associated color map address points to the color map entry that has been changed.

Color mode 0 makes use of the color map by establishing the in-use color map address to be the existing map address of the color value specified in the color mode 0 set color command. If that color is not already specified in the color map, color mode 0 makes use of the lo address which has not been specified since the l reset command (reestablishing the default color map) and w is not the address of nominal black or nominal whi If no addresses are available the color map will not changed and the in-use drawing color is established in an imp dependent manner.

The following relation between the number of bits (N) in the color map address and the number of bits (M) in the color values stored in the color map is recommended.

$$M \geq 3\,(N-1)$$

The SET COLOR opcode takes a multi-value operand and is shown in Figures 22 and 23. The color value operand is used to define a color according to Figure 23.

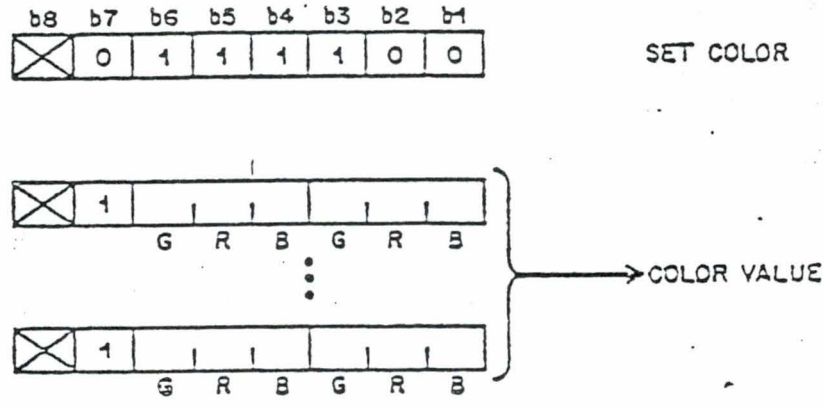


Figure 22

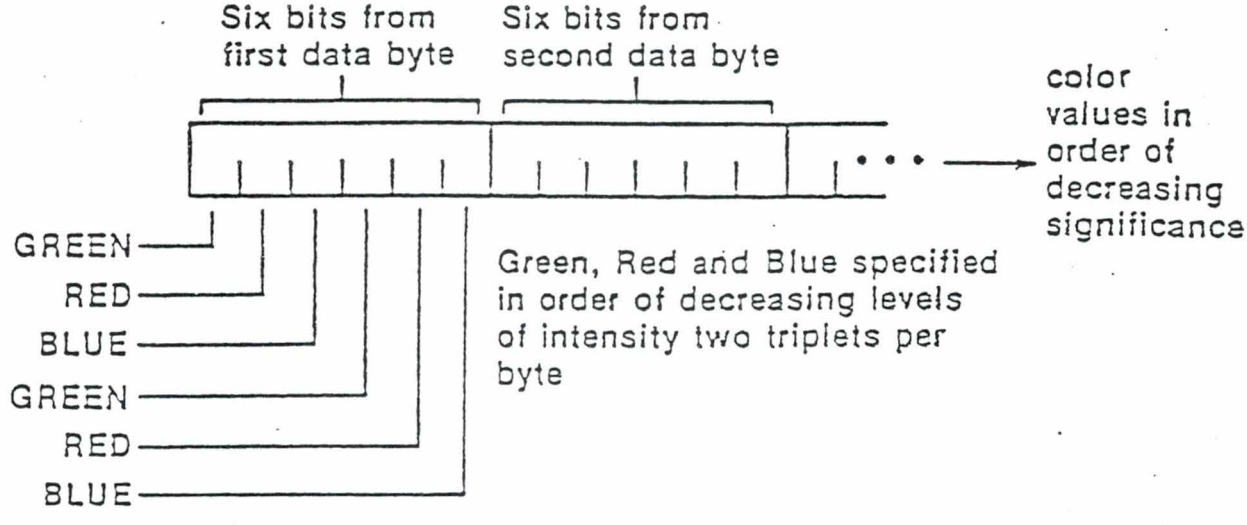


Figure 23

Color Encoding Scheme

In color mode 0, this sets the in-use drawing color. This color remains in-use and is applied to subsequently received alphanumerics and pictorial drawings until changed by either another SET COLOR command, the RESET command, described in Clause 5.2.3.2.9 or the NSR control character, described in Clause 6.1.6.5 The default drawing color in color mode 0 is white. A background color cannot be specified in color mode 0, that is, character patterns and pictorial drawings merely overwrite the existing contents of the display storage medium.

In color modes 1 and 2, the SET COLOR command is used to load color values into the color map. The address of the entry to be loaded is taken to be the one indicated by the in-use drawing color (which must have been set previously with the SELECT COLOR command).

If the maximum size entry (i.e., number of bits) that the color map can accommodate is smaller than the number of bits provided by the SET COLOR operand, the operand is truncated and only the most significant bits are used. If the maximum size entry that the color map can accommodate is larger than the number of bits provided by the SET COLOR operand, the operand is padded with zeroes. The maximum color fraction attainable given the number of bits specified in the color value operand shall be interpreted as full saturation.

color map address associated with the

If the SET COLOR command is implicitly repeated via the sending of additional numeric data, the address of the color entry to be changed is automatically incremented prior to the execution of the new opcode. This does not affect the in-use drawing color.

If no operand follows a SET COLOR opcode, the transparent color is set. If transparent color is selected, then any lower order planes would show through the display. (These lower order planes could correspond to planes with a lower Z value in a multi-planar terminal architecture or an analog video signal in applications where the videotex display is superimposed over a standard television image, e.g., for captioning.) If there are no lower order planes or if transparency is not implemented, then the transparent color shows as black.

The default contents of the color map are defined according to the algorithm described below:

N = number of bits in the color map address

2^N = size of the color map

M = number of bits in the color values (i.e., width of the color map)

M ≥ 3(N-1) as specified previously

The algorithm for incrementing is to change the most significant 0 to a 1 o to change all 1's to the left of it to 0 for example, color map address 010100 incremented first to 110100 then to 001

The first half of the default color map is used to store a complete, uniformly spaced grey scale. This comprises the ordered set of colors where R = G = B. (Note that if M = 3 (N-1), there should be exactly $(2^N)/2$ grey levels including black and white.) The second half of the default color map is used to store a full range of hues equally spaced around the perimeter of the hue circle. The hue circle is shown in Figure 24 and is defined with the three primary colors (red, green, and blue) lying equidistant around the circle with blue at 0°, red at 120°, and green at 240°. All other hues can be obtained with various combinations of these three primaries mixed in proportions that are a function of the position of the desired hue on the hue circle. The algorithm for obtaining the RGB values for the default hues, which lie equally spaced around the hue circle starting at 0° and proceeding counter-clockwise, is as follows:

Let

h = desired hue

ang h = the angle of h

P_1 = the closest primary to h

ang P_1 = the angle of P_1

P_2 = the second closest primary to h

ang P_2 = the angle of P_2

P_3 = the furthest primary from h

The values of the primaries in the RGB system that must be combined to give the hue h will be:

1) $P_1 = 1$ (i.e., all bits set 1)

2) $$P_2 = \frac{|\text{ang h} - \text{ang } P_1|}{60°}$$

3) $P_3 = 0$ (i.e., all bits set 0)

Table 15 shows an example of a default color map for N = 4 and M = 9.

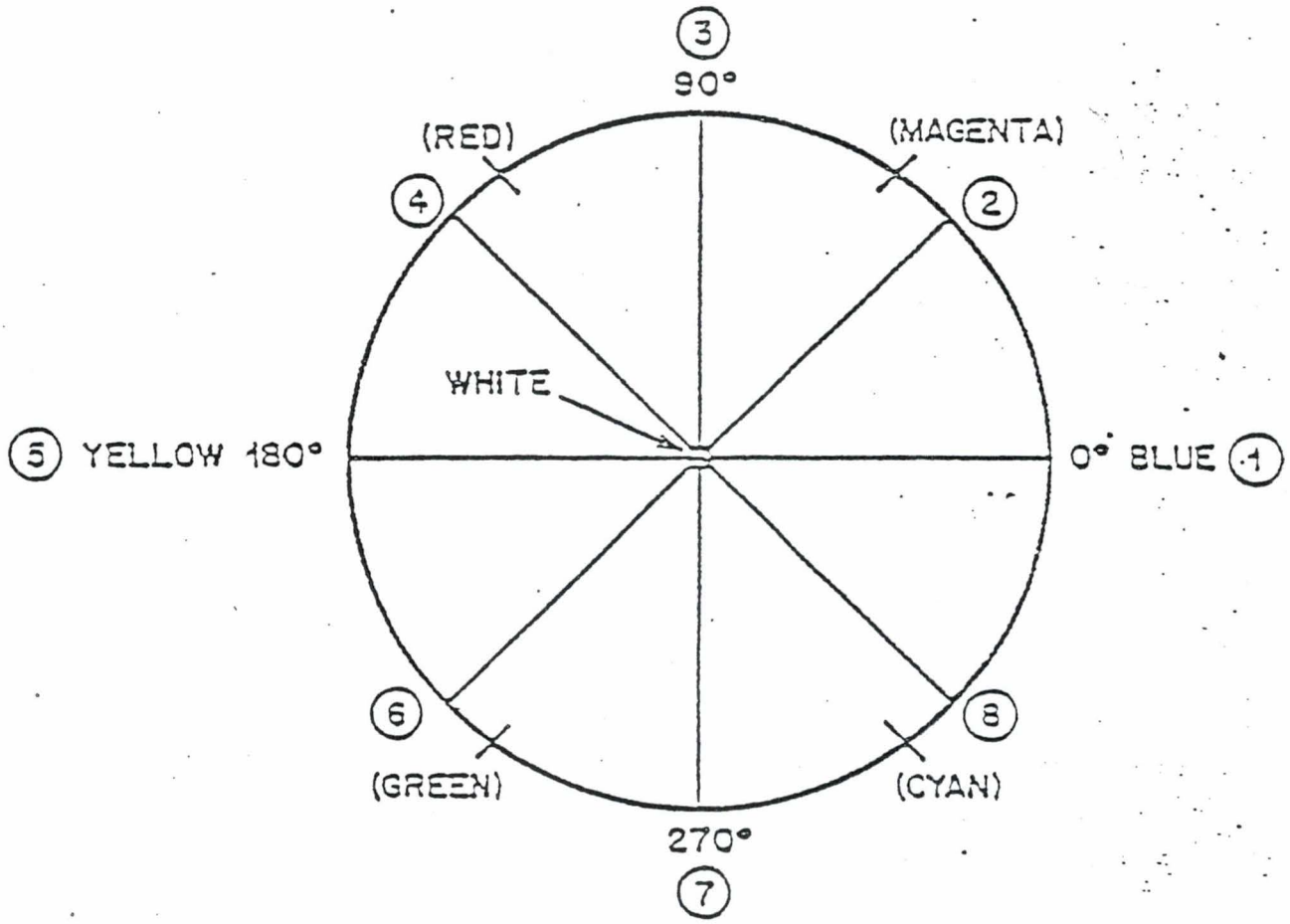


Figure 24
Selection of Default Colors

Table 15

COLOR MAP ADDRESS	COLOR VALUES		
	R	G	B
0 0 0 0	0 0 0	0 0 0	0 0 0
0 0 0 1	0 0 1	0 0 1	0 0 1
0 0 1 0	0 1 0	0 1 0	0 1 0
0 0 1 1	0 1 1	0 1 1	0 1 1
0 1 0 0	1 0 0	1 0 0	1 0 0
0 1 0 1	1 0 1	1 0 1	1 0 1
0 1 1 0	1 1 0	1 1 0	1 1 0
0 1 1 1	1 1 1	1 1 1	1 1 1
1 0 0 0	0 0 0	0 0 0	1 1 1
1 0 0 1	1 0 1	0 0 0	1 1 1
1 0 1 0	1 1 1	0 0 0	1 0 0
1 0 1 1	1 1 1	0 1 0	0 0 0
1 1 0 0	1 1 1	1 1 1	0 0 0
1 1 0 1	0 1 0	1 1 1	0 0 0
1 1 1 0	0 0 0	1 1 1	1 0 0
1 1 1 1	0 0 0	1 0 1	1 1 1

Default Color Map for N = 4, M = 9

5.3.2.6 SELECT COLOR

5.3.2.6.1 The SELECT COLOR opcode is used to set the color mode as well as select the in-use color for mode 1 and 2. (See Figure 25.)

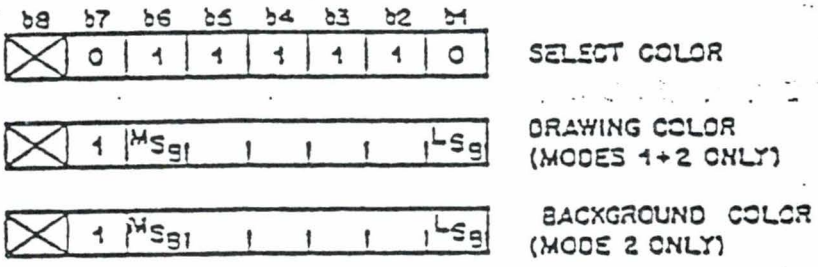


Figure 25

The SELECT COLOR opcode can take zero, one, or two single-value operands. Any additional numeric data is ignored. If the SELECT COLOR opcode is followed by no operands, color mode 0 is indicated. ~~(On terminals with implicitly defined color maps, this will also reinitialize the internal color map.)~~ The terminal will remain in color mode 0 until either another SELECT COLOR command with operands is received or the color mode is changed with a RESET command (described in Clause 5.3.2.9). While in color mode 0, the SET COLOR command is used to set the in-use drawing color, as described in the previous section, and the SELECT COLOR command is not used.

If the number of bits of color map address (N) implemented is less than the number of concatenated bits available in a single-value operand (6, 12, 18 or 24 depending on the single-value length operand of the most recently received DOMAIN command or the default of 6 if no DOMAIN command has been received since a RESET or NSR command), only the high order bits are significant. In other words, the number of bits required to specify the color map address is left justified within the single-value operand. For example, for the default single-value operand length of one byte and a four bit color map address (N=4), the receiving presentation process responds to b8 through b5 and ignores b4 and b3 of each color map address operand in the SET COLOR and BLINK command. If the number of bits of color map address (N) implemented is greater than the number of concatenated bits available in a single-value operand (6, 12, 18 or 24), trailing zero bits are supplied by the receiving presentation process.

5.3.2.6.2 If the SELECT COLOR opcode is followed by a single operand, color mode 1 is indicated. (This has no effect on the color map.) The terminal will remain in color mode 1 until either another SELECT COLOR command with 0 or 2 operands is received or the color mode is changed with a RESET or NSR command. While in color mode 1, the single operand following the SELECT COLOR opcode is used to set the in-use drawing color that is applied to subsequently received alphanumeric text and pictorial information. Note, again, that the in-use drawing color in this case is an ordinal number that represents an address in the color map in which the actual color value was previously, or will later be, loaded with a SET COLOR command. A background color is not specified in color mode 1, rather, alphanumerics and pictorial drawings merely overwrite the existing contents of the display storage medium only where the in-use color is applied.

5.3.2.6.3 If the SELECT COLOR opcode is followed by two operands, color mode 2 is indicated. Again, the terminal will remain in color mode 2 until either another SELECT COLOR command with 0 or 1 operands is received or the color mode is changed with a RESET or NSR command. While in color mode 2, the first operand following the SELECT COLOR opcode is used to set the in-use drawing color and the second operand is used to set the in-use background color. Characters received while in color mode 2 will be drawn in the in-use drawing color over the in-use background color, which occupies the remainder of the character field. For the special case in which the two operands are identical, i.e., the in-use drawing color is specified to be the same as the in-use background color, the in-use drawing color is, instead, left at its current value and only the in-use background color is changed to the value specified. The in-use background color also applies to the highlight as well as the alternating color in the line and area texture patterns, as described in Clause 5.3.2.4.

5.3.2.7 BLINK

5.3.2.7.1 The BLINK opcode is used to cause a color map entry to periodically alternate between two colors.

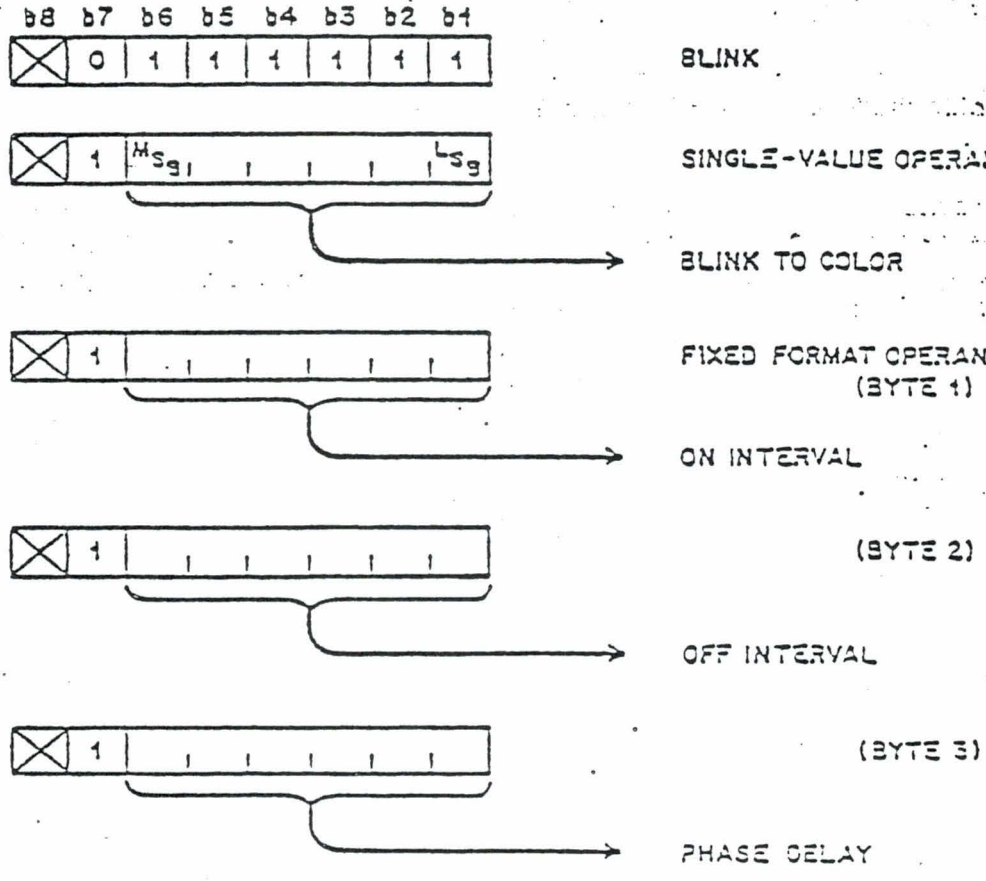


Figure 26

5.3.2.7.2 The mechanism for performing this is a blink process. This process periodically overwrites the contents of the current in-use drawing color (the "blink-from" color) and substitutes the current contents of another entry in the color map, which is called the "blink-to" color. The blink-to color is activated for a period of time known as the ON interval. The blink-from color is activated for a period of time known as the OFF interval. The ON and OFF intervals alternate with each other. A phase delay may also be specified, this being a delay in the start of the ON interval referenced to the start of the ON interval of the last active blink process defined. If multiple blink processes have ON or OFF intervals that expire simultaneously, they are processed sequentially starting with the most recently defined blink process and ending with the least recently defined blink process. In this case, each blink process takes as its input the color map that resulted from the previously executed blink process.

5.3.2.7.3 The first single-value operand following the BLINK opcode is the blink-to color specification, specified as a color map address (see Clause 5.3.2.6.1). The next fixed format operand is the ON interval specified in units of 1/10 of a second. Only bits b6 through b1 are used for this specification. In a similar manner, the next fixed format operand specifies the OFF interval. The fourth fixed format operand specifies the phase delay, also in units of 1/10 of a second. If this byte is omitted, a phase delay of 0 is indicated, and if there are no currently active blink processes, it is ignored. An ON or OFF interval of 0 is taken to mean termination of any active blink process on the blink-from/blink-to color pair (see Figure 26).

5.3.2.7.4 Defining a blink process on a pair of blink-to and blink-from colors automatically terminates any previously defined blink process operating on the same pair of colors. If no operands follow the blink opcode, then all blink processes utilizing the current in-use color as the blink-from color will be terminated. The original blink-from color should be restored (unless it has been changed explicitly by a SET COLOR command) when all blink processes using that blink-from color are terminated.

5.3.2.7.5 If additional data follow a completely specified blink process, then the BLINK command is implicitly repeated, with the address of the blink-from color being automatically incremented prior to the execution of the new opcode. The in-use drawing color is not affected by this incrementing.

5.3.2.8 WAIT

5.3.2.8.1 The WAIT command is used to cause a delay in processing for a specific time interval.

The wait interval starts at the completion of the execution of the command preceding WAIT or the receipt of the WAIT command, whichever occurs later.

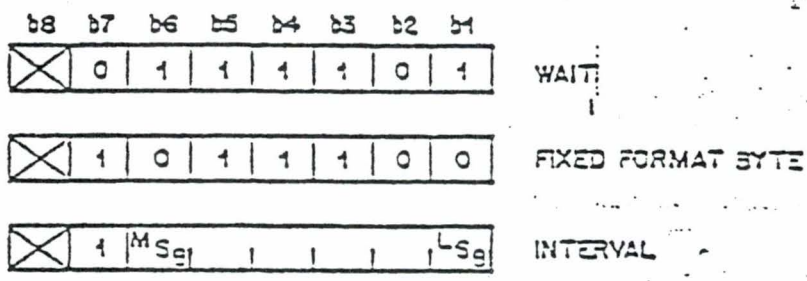


Figure 27

5.3.2.8.2 The first byte of operand data following the WAIT opcode must follow the format given in Figure 27. The next byte of operand data gives the time delay in units of 1/10 of a second (64 binary coded values). Only bits b1 through b6 are used for this purpose. If any additional data bytes follow, they are treated as additional periods of waiting time, each period being specified independently by each data byte. An operand of zero indicates a wait interval for a minimum duration that is implementation dependent.

5.3.2.9 RESET

5.3.2.9.1 The RESET command is used to selectively reinitialize the control and attribute parameters to their default values, clear the screen, set the border color, home the cursor, and clear the DRCS set, texture attributes, macros, and unprotected fields (described in Clause 6.2). The RESET opcode takes a two byte, fixed format operand. The order of execution of the resets is byte 1, low order bit (b1) to high order bit (b6), followed by byte 2, low order bit (b1) to high order bit (b6). The RESET opcode and its operand are shown below. (See Figure 28.)

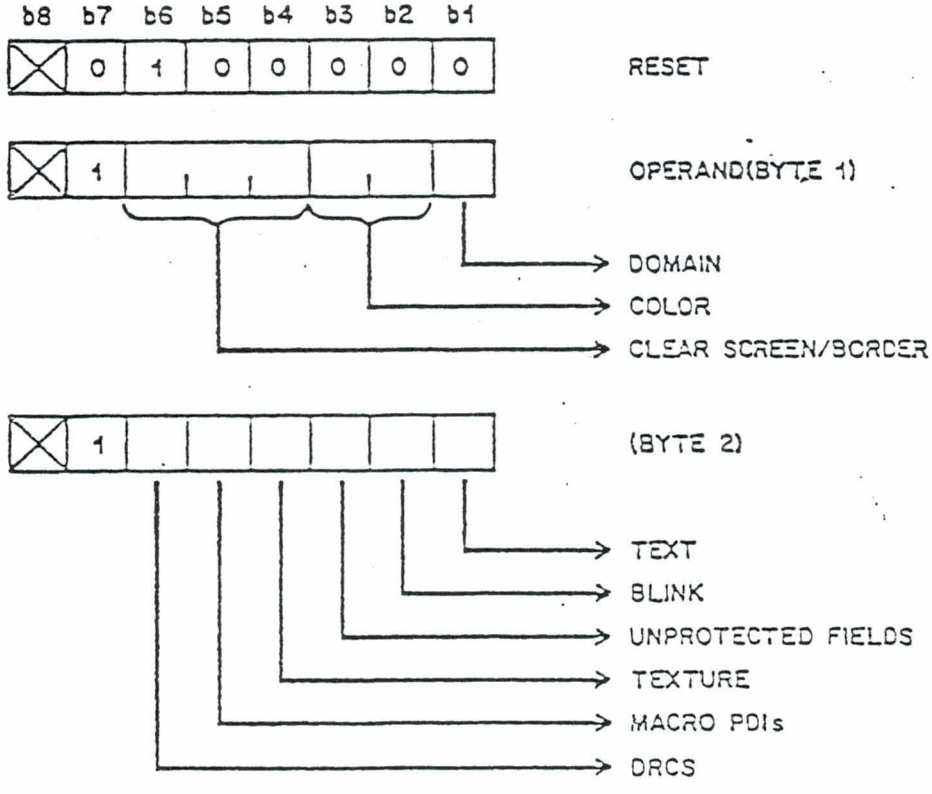


Figure 23

5.3.2.9.2 Operand Bytes

5.3.2.9.2.1 Operand Byte 1 of RESET. If bit 1 of byte 1 equals 1, the DOMAIN parameters are reset to their default values. If b1 is 0, the DOMAIN parameters are unchanged.

Bits 2 and 3 of byte 1 modify the color mode and/or current drawing color as shown in Table 16.

Table 16

b3	b2	Color mode
0	0	No action.
0	1	Select color mode 0, ~~initialize implicit color map if it exists.~~
1	0	Select color mode 1 and set color map to default colors.
1	1	Select color mode 1, set color map to default colors, and set the in-use drawing color to white.

Bits 4, 5, and 6 of byte 1 clear the ~~screen~~ physical display area and/or border area to the colors shown in Table 17.

The border area surrounds the physical display area and may only be set to one color at a time. ~~(Screen and physical display area are synonymous.)~~

Table 17

b6	b5	b4	Color
0	0	0	No action.
0	0	1	Physical display area ~~Screen~~ to black.
0	1	0	Physical display area ~~Screen~~ to current drawing color.
0	1	1	Border to black.
1	0	0	Border area to current drawing color.
1	0	1	Physical display area ~~Screen~~ and border to current drawing color.
1	1	0	Physical display area ~~Screen~~ to current drawing color and border area to black.
1	1	1	Physical display area ~~Screen~~ and border area to black.

5.3.2.9.2.2 Operand Byte 2 of RESET. If bit b1 of byte 2 equals 1, the cursor is sent to its home position (top left character position on the screen) and all text parameters (from the TEXT opcode, from the C1 set and the ~~active Drawing Area~~ active fi) are reset to their default values. If b1 is 0, the text parameters and the cursor position are unchanged.

If bit b2 of byte 2 equals 1, all blink processes are terminated. If b2 is 0, then blink processes are not affected.

If bit b3 of byte 2 equals 1, all unprotected fields are changed to protected status but the contents are unaffected. If b3 is 0, unprotected fields are left unaffected. (However, the field definitions (except tho of the active field) are lost, as well as any data structures maintained for user editing a transmiss
If bit b4 of byte 2 equals 1, all texture attributes are set to their default values. The four programmable texture masks are not cleared. If b4 is 0, current texture attributes are not changed.

If bit b5 of byte 2 equals 1, all macros are cleared. This includes transmit-macros. If b5 is 0, macros are unaffected.

If bit b6 of byte 2 equals 1, all DRCS characters are cleared, that is, all character positions are set to the space character. If b6 is 0, the DRCS characters are unaffected.

If the RESET command is received with no operands, it is interpreted as if it had been sent with all bits in both bytes set equal to 1. If only the first byte is received, the second byte is then interpreted as if it had been received with all zeros. If more than two data bytes are received, the additional bytes are ignored.

For descriptions of transmit-macro and DRCS, see Clauses 5.5 and 5.6 respectively.

5.3.3 Geometric Drawing Primitives

5.3.3.1 POINT

5.3.3.1.1 The POINT opcode is used to perform the two most basic geometric drawing operations, that of establishing the coordinate at which to commence drawing and that of drawing a point. A coordinate pair must always be specified with this command to set the drawing position (SET). Optionally, a dot may be drawn (i.e., made visible) at the specified coordinate position (POINT). The coordinate may be specified either as an absolute (X,Y) position or as a relative (dx,dy) displacement from the previous drawing position. (See Figure 29.)

A series of coordinate positions following a POINT opcode may be used to draw a point by point graph.

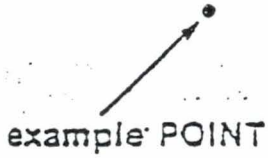


Figure 29

POINT

5.3.3.1.2 Commands

5.3.3.1.2.1 General. The commands used are described in Clauses 5.3.3.1.2.2 to 5.3.3.1.2.5.

5.3.3.1.2.2 POINT SET (Absolute, Invisible). This opcode sets the drawing position to the absolute coordinates specified. A dot is not drawn. (See Figure 30.)

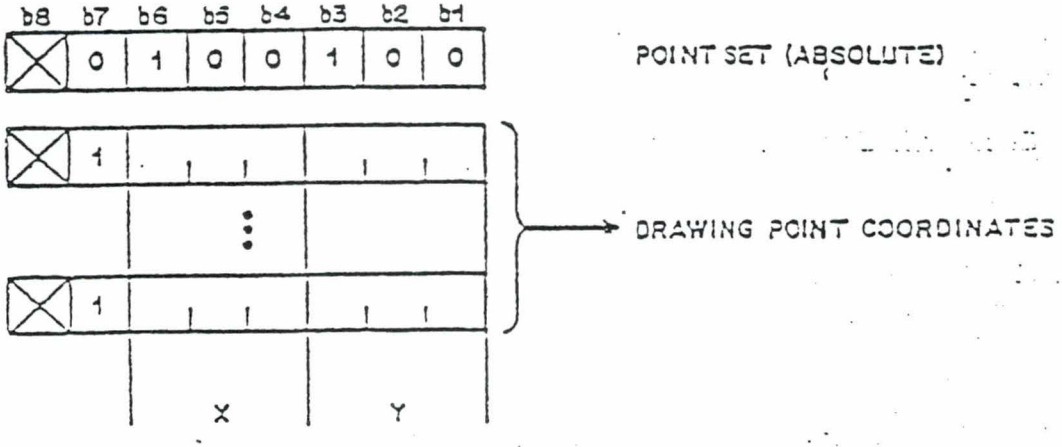


Figure 30

POINT SET (Absolute, Invisible)

5.3.3.1.2.3 POINT SET (Relative, Invisible). This opcode sets the drawing position to the coordinates obtained by adding the displacement specified to the current drawing position. A dot is not drawn. (See Figure 31.)

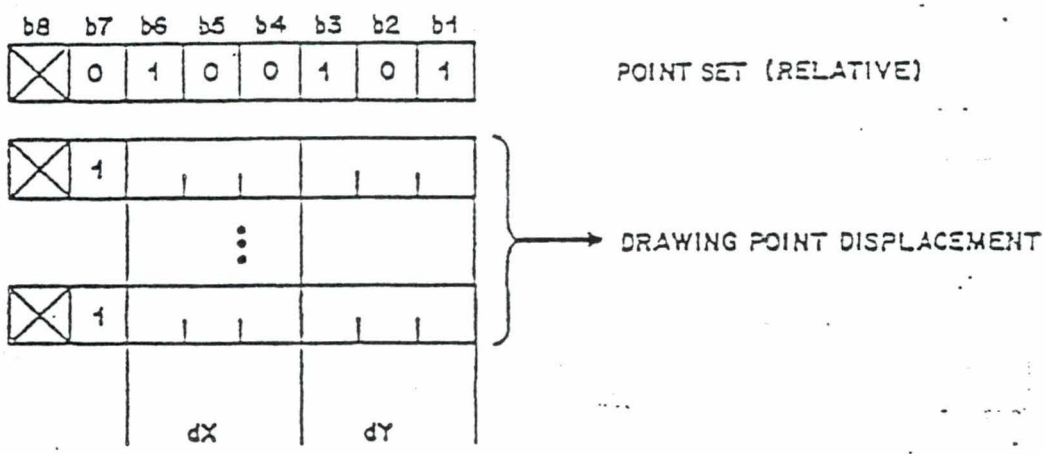


Figure 31

POINT SET (Relative, Invisible)

5.3.3.1.2.4 POINT (Absolute, Visible). This opcode sets the drawing position to the absolute coordinates specified and draws a dot at that point whose size is determined by the logical pel size, and whose color is determined by the in-use color. (See Clause 5.3.2.2 for a description of logical pel.) (See Figure 32.)

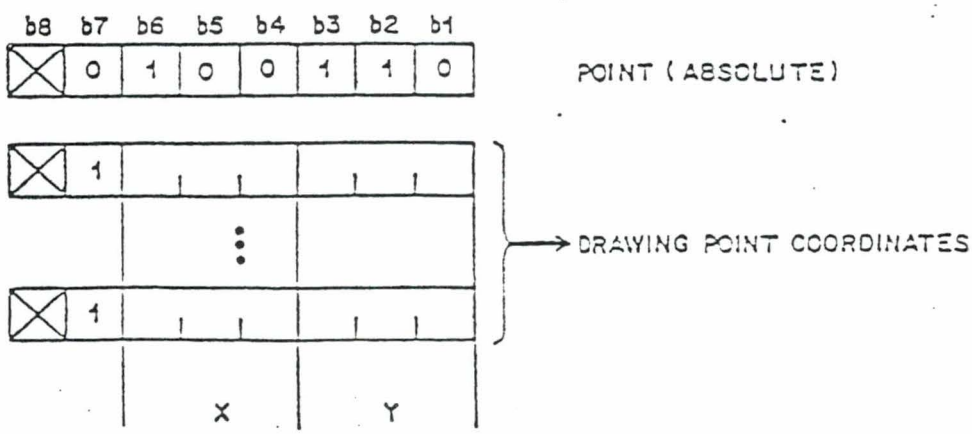


Figure 32

POINT (Absolute, Visible)

5.3.3.1.2.5 POINT (Relative, Visible). This opcode sets the drawing position to the coordinates obtained by adding the displacement specified to the current drawing position, and draws a dot at that point whose size is determined by the logical pel size, and whose color is determined by the in-use color. (See Figure 33.)

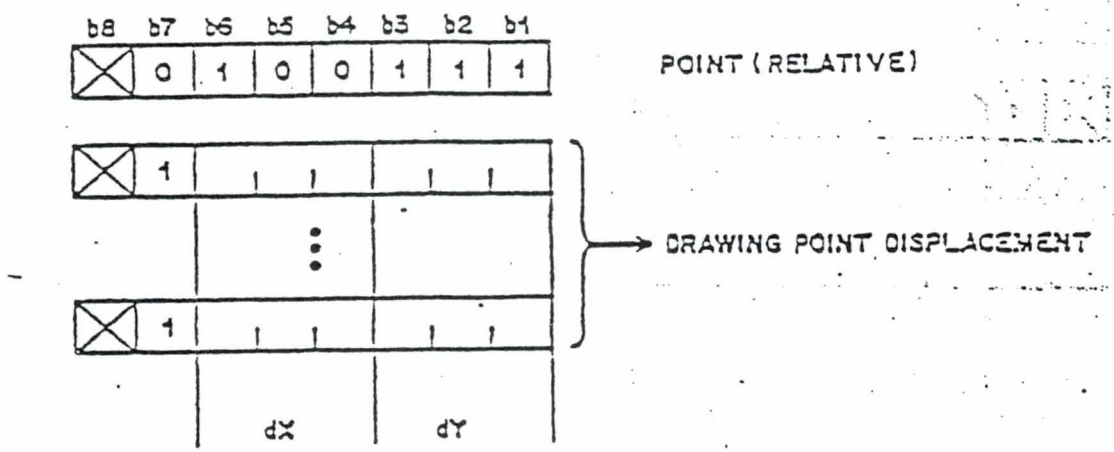


Figure 33

POINT (Relative, Visible)

5.3.3.2 LINE

5.3.3.2.1 The LINE command is the second most basic geometric drawing operation. The direction and length of a line are specified by the endpoints. The initial drawing position of a line drawing operation may be either explicitly specified within the LINE opcode or interpreted as being the final position of the previous drawing opcode. The final drawing position for a line segment may be specified either as a relative (dx,dy) displacement from the initial position or as an absolute (X,Y) coordinate. The line is drawn in the in-use color(s), has a width that is determined by the logical pel size, and a texture determined by the line current texture attribute. (See Figure 34.)

The LINE opcode may be used to draw a line graph from a table of numbers described as absolute or relative coordinates in the same manner as the POINT opcode.

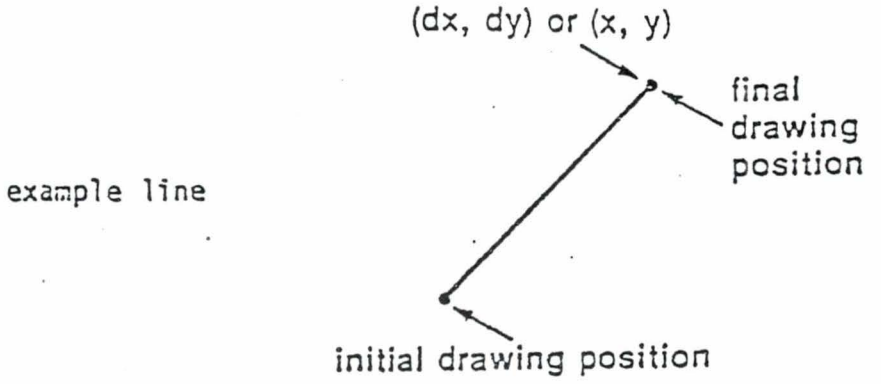


Figure 34
LINE

5.3.3.2.2 Commands

5.3.3.2.2.1 General. The commands used are described in Clauses 5.3.3.2.2.2 to 5.3.3.2.2.5.

5.3.3.2.2.2 LINE (Absolute). This opcode draws a line between the current drawing point and the end-point, which is specified in absolute coordinates. (See Figure 35.)

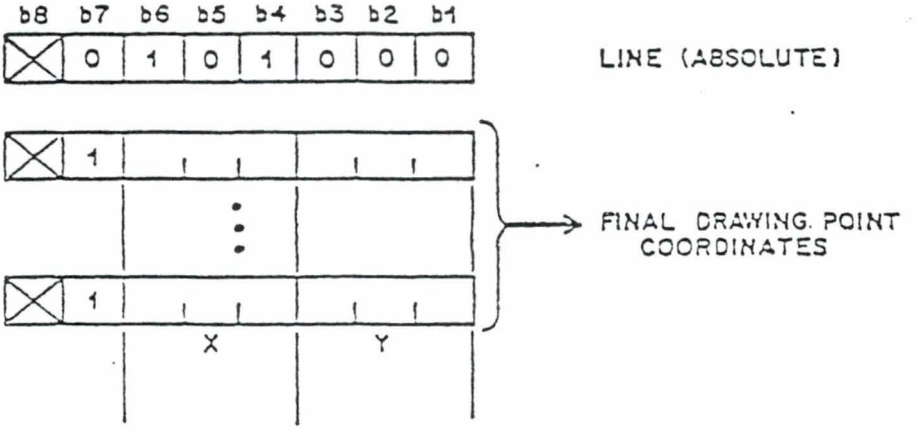


Figure 35
LINE (Absolute)

5.3.3.2.2.3 LINE (Relative). This opcode draws a line between the current drawing point and the endpoint, which is specified as a relative displacement to the current drawing point. (See Figure 36.)

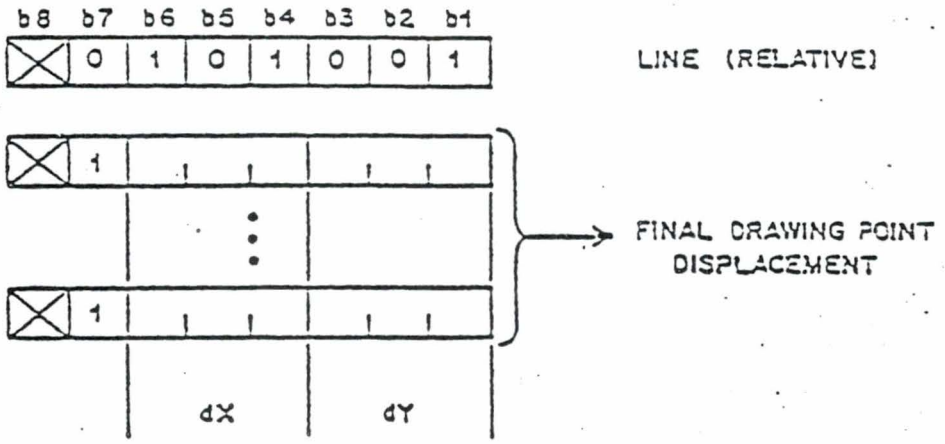


Figure 36

LINE (Relative)

5.3.3.2.2.4 SET and LINE (Absolute). This opcode draws a line between the initial drawing point coordinates and the final drawing point coordinates, both of which are specified in absolute coordinates. (See Figure 37.)

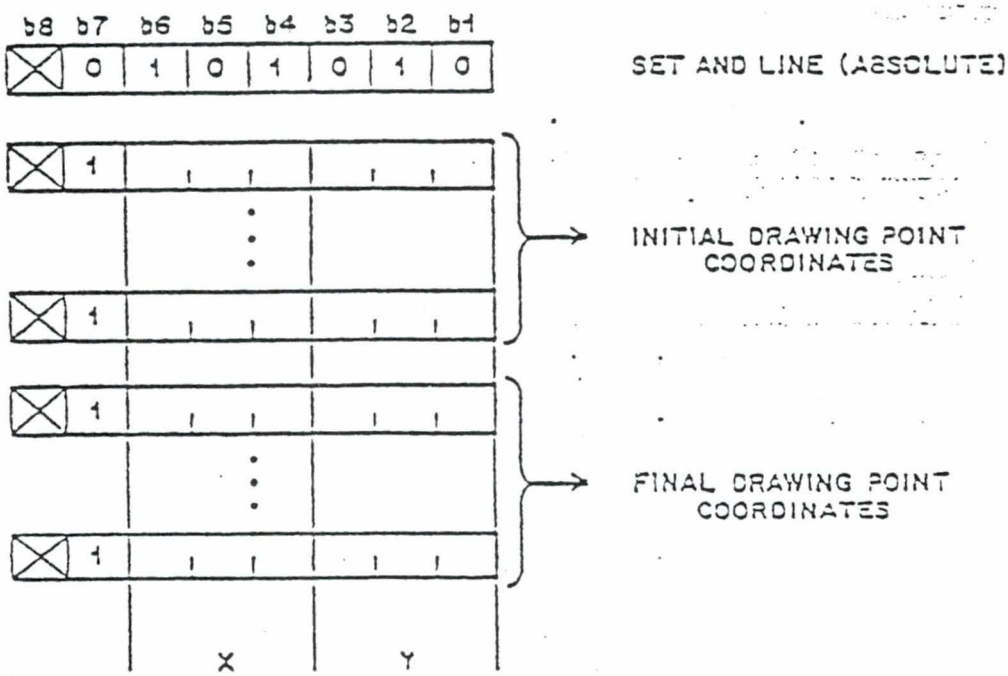


Figure 37

SET and LINE (Absolute)

5.3.3.2.2.5 SET and LINE (Relative). This opcode draws a line between the initial drawing point coordinates, which are specified absolutely, and the final position, which is specified as a relative displacement to the first set of coordinates. (See Figure 38.)

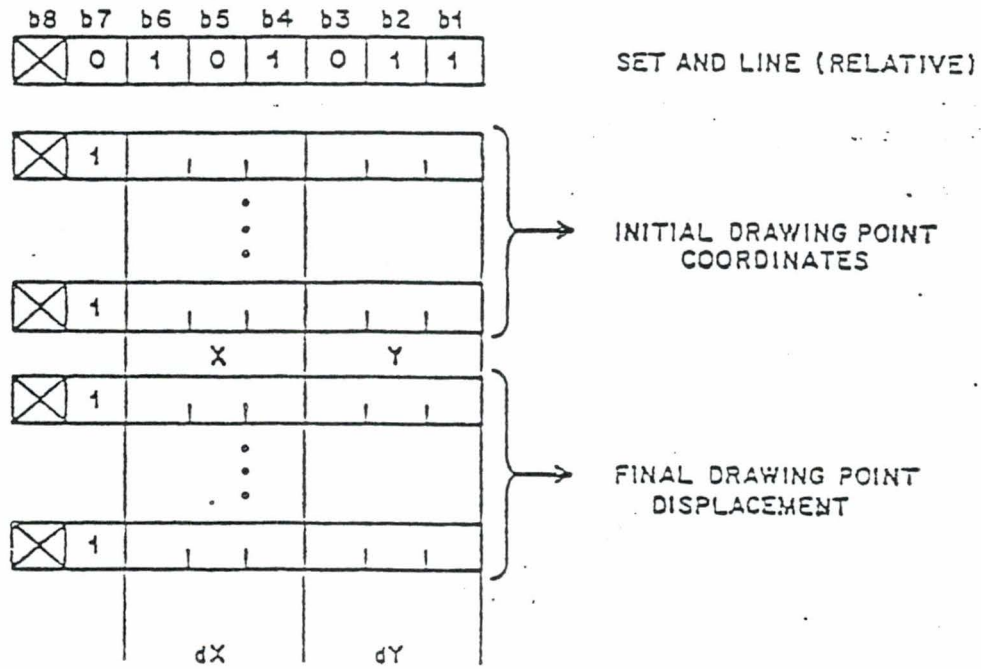


Figure 38

SET and LINE (Relative)

5.3.3.3 ARC

5.3.3.3.1 The ARC geometric drawing operation provides the capability of drawing circles, segments of circles, and curvilinear splines. An arc is drawn from an initial drawing position to a final drawing position through an intermediate point on the arc. The initial drawing position may be either explicitly specified within the ARC opcode or interpreted as being the final position of the previous drawing opcode. The intermediate position on the arc is described as a relative displacement from the initial drawing position. The final drawing position is specified as a relative displacement from the intermediate position on the arc. It is good practice, in order to minimize error, to always specify the intermediate point on the arc as being approximately midway between the start and end points.

The current drawing position at the completion of drawing an arc is the endpoint specified. Drawing a circle results when the start and endpoint are coincident. For the definition of a circle, the point on the arc defines the diameter of the circle and therefore is the midpoint between the start and endpoint. If the three drawing points are co-linear, a line is drawn from the initial point to the final point, except for the error condition in which the intermediate point does not lie between the initial and final points. If the endpoint is omitted, it is taken to be coincident with the start point and a circle is drawn. Note that the arc may not be specified so that any portion of it lies outside the unit screen (see Clause 5.3.1.1).

If more data are given after the endpoint specification, a curvilinear spline is drawn through the points specified. The exact algorithm for the curvilinear spline is reserved for future study.

The arc is drawn in the in-use color(s), has a width that is determined by the logical pel size, and has a texture given by the current line texture attribute (see Clause 5.3.2.4).

The area enclosed by an ARC opcode may be filled in the texture pattern as defined by the current texture attribute. The form of the filled-in area is the area enclosed by the ARC command and the chord joining the two endpoints of the arc. The chord is not considered a part of the arc and, as such, is not highlighted if highlight mode is selected (see Clause 5.3.2.4 for a discussion of highlight mode). (See Figure 39.)

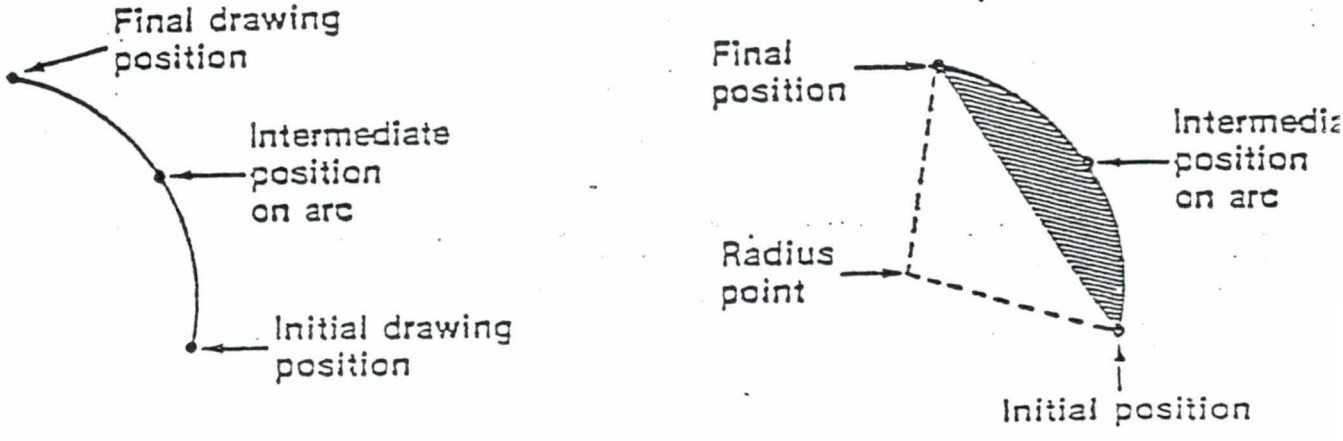


Figure 39

ARC

5.3.3.3.2 Commands

5.3.3.3.2.1 General. The commands used are described in Clauses 5.3.3.3.2.2 to 5.3.3.3.2.5.

5.3.3.3.2.2 ARC (Outlined). This opcode causes an arc to be drawn through three points. The initial point position is the current drawing point, the intermediate point position is the first block of coordinate data, specified as a relative displacement from the initial point, and the final point position is the second block of coordinate data, specified as a relative displacement from the intermediate point. The arc is not filled. (See Figure 40.)

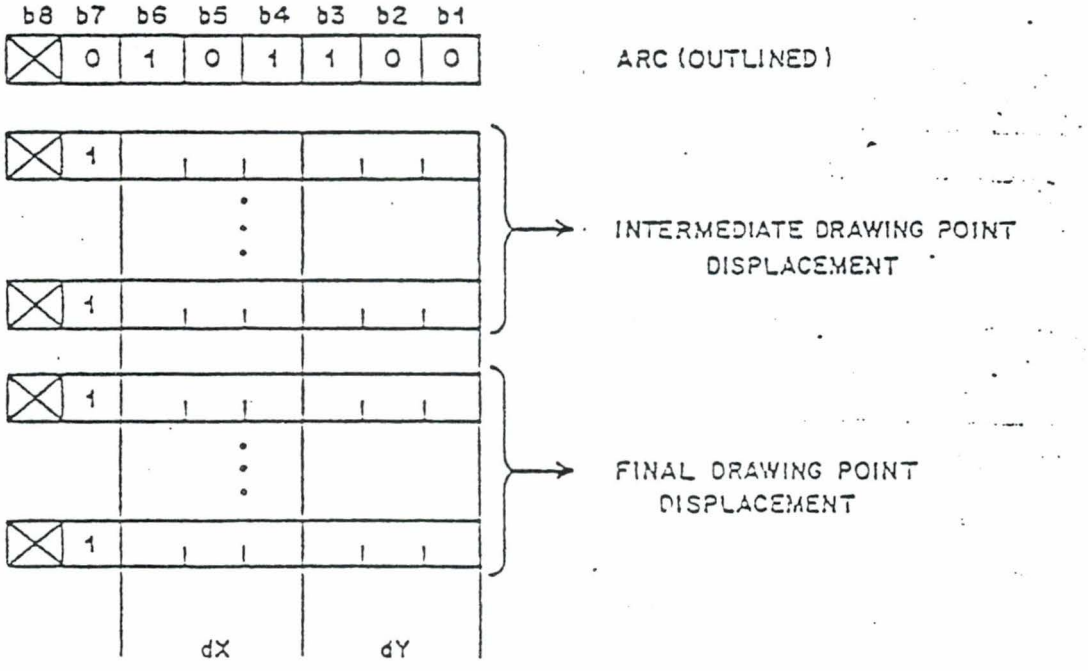


Figure 40
ARC (Outlined)

5.3.3.3.2.3 ARC (Filled). This opcode causes an arc to be drawn through three points. The initial point position is the current drawing point, the intermediate point position is the first block of coordinate data, specified as a relative displacement from the initial point, and the final point position is the second block of coordinate data, specified as a relative displacement from the intermediate point. The initial and final drawing positions are joined by a chord and the resulting figure is filled in the current color(s) with the current texture pattern. (See Figure 41.)

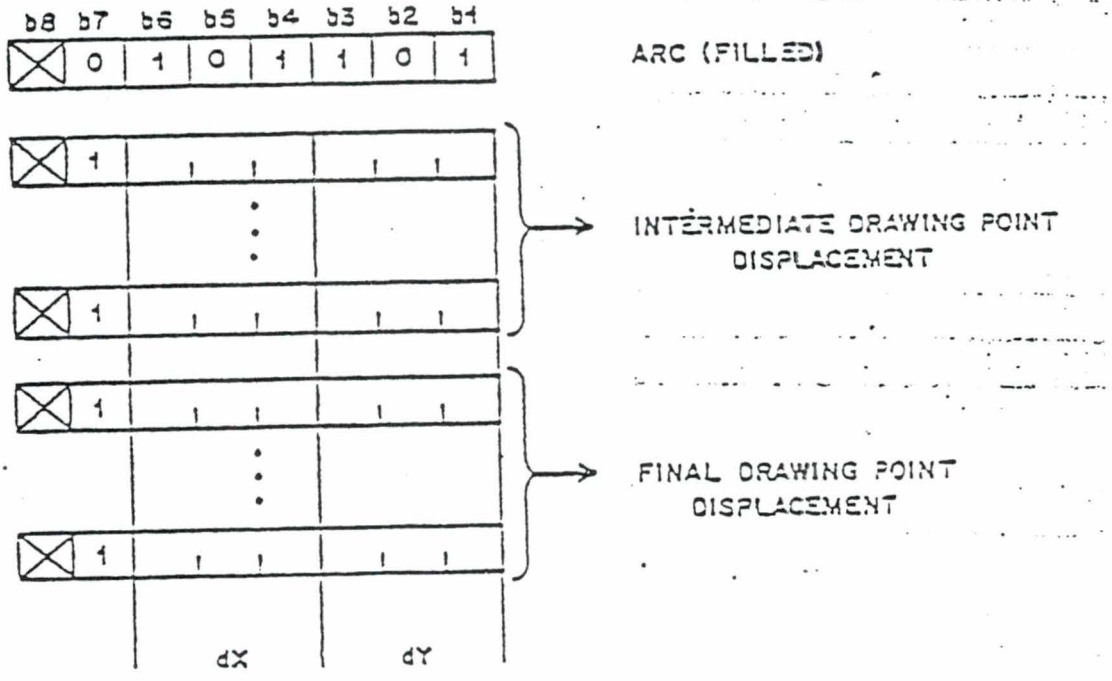


Figure 41
ARC (Filled)

5.3.3.3.2.4 SET and ARC (Outlined). This opcode causes an arc to be drawn through three points. The initial point position is the first block of coordinate data, specified in absolute coordinates. The intermediate point position is the second block of coordinate data, specified as a relative displacement from the initial point, and the final point position is the third block of coordinate data, specified as a relative displacement from the intermediate point. The arc is not filled. (See Figure 42.)

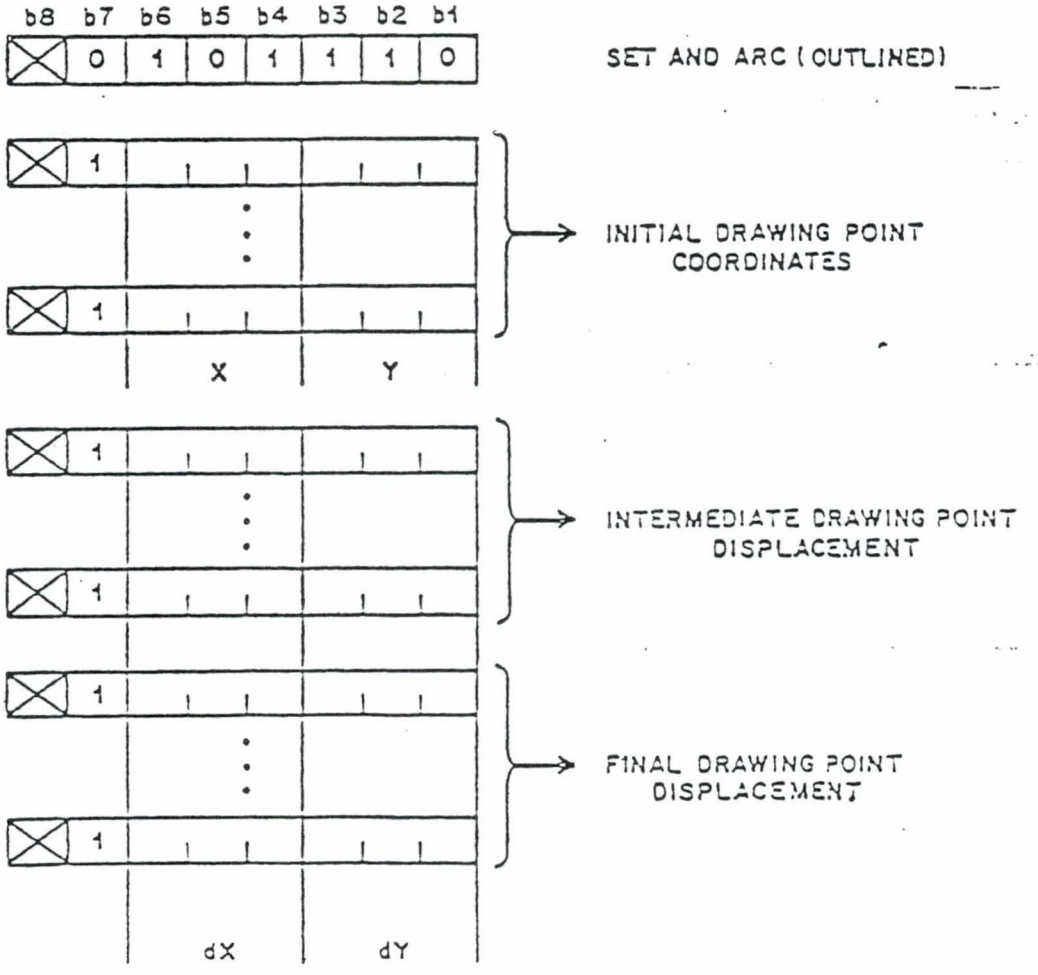


Figure 42

SET and ARC (Outlined)

5.3.3.3.2.5 SET and ARC (Filled). This opcode causes an arc to be drawn through three points. The initial point position is the first block of coordinate data, specified in absolute coordinates. The intermediate point position is the second block of coordinate data, specified as a relative displacement from the initial point, and the final point position is the third block of coordinate data, specified as a relative displacement from the intermediate point. The initial and final drawing positions are joined by a chord and the resulting figure is filled in the current color(s) with the current texture pattern. (See Figure 43.)

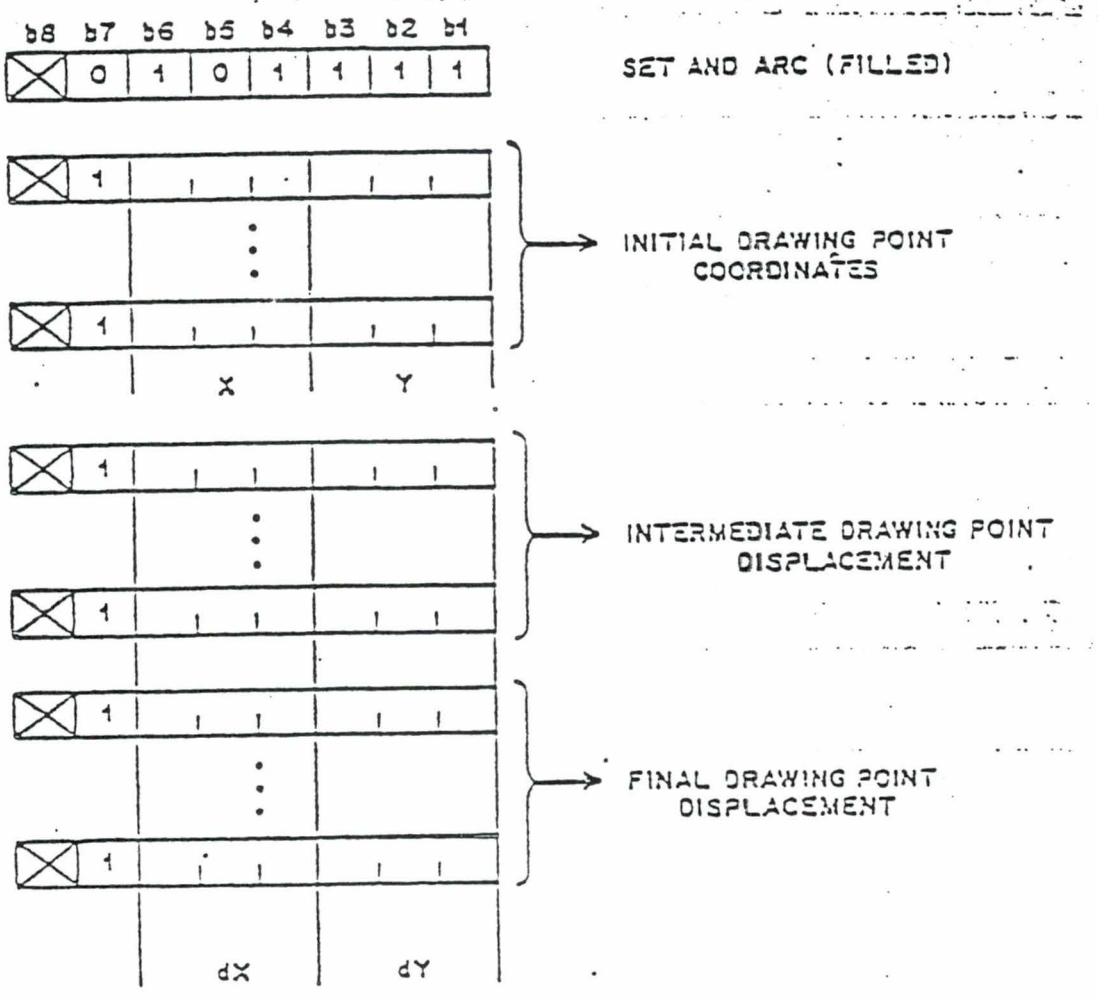


Figure 43

SET and ARC (Filled)

5.3.3.4 RECTANGLE

5.3.3.4.1 The RECTANGLE geometric drawing operation provides the capability of drawing a rectangular area of width dx and height dy. The initial drawing position of a RECTANGLE drawing operation may be either explicitly specified within the RECTANGLE opcode or interpreted as being the final position of the previous drawing opcode. The final drawing position of a RECTANGLE opcode is the initial drawing position altered in x only, by the amount of the dx displacement.

A rectangle may be either filled or outlined. Outlined rectangles are drawn in the in-use color(s) and have a line width that is determined by the logical pel size and a line texture that is specified by the TEXTURE command. For filled rectangles, the area is filled in the current color(s) with the texture pattern specified in the TEXTURE command, and the perimeter may be optionally highlighted.

The RECTANGLE opcode may be used to draw a histogram from a table of numbers representing relative dy and dx displacements in the same manner as the POINT and LINE opcode graph plot mode. (See Figure 44.)

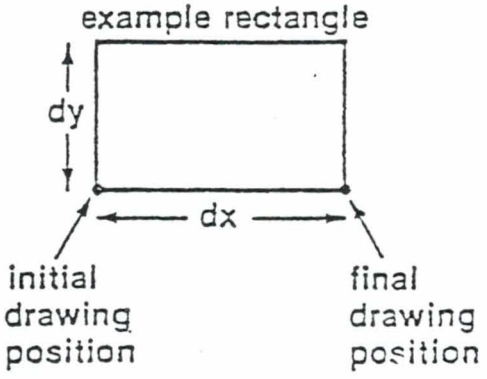


Figure 44
RECTANGLE

5.3.3.4.2 Commands

5.3.3.4.2.1 General. The commands used are described in Clauses 5.3.3.4.2.2 to 5.3.3.4.2.5.

5.3.3.4.2.2 RECTANGLE (Outlined). This opcode causes a rectangle to be drawn. The initial drawing position is the current drawing point and the width and height (dx,dy) are given as the first block of coordinate data. The rectangle is not filled. (See Figure 45.)

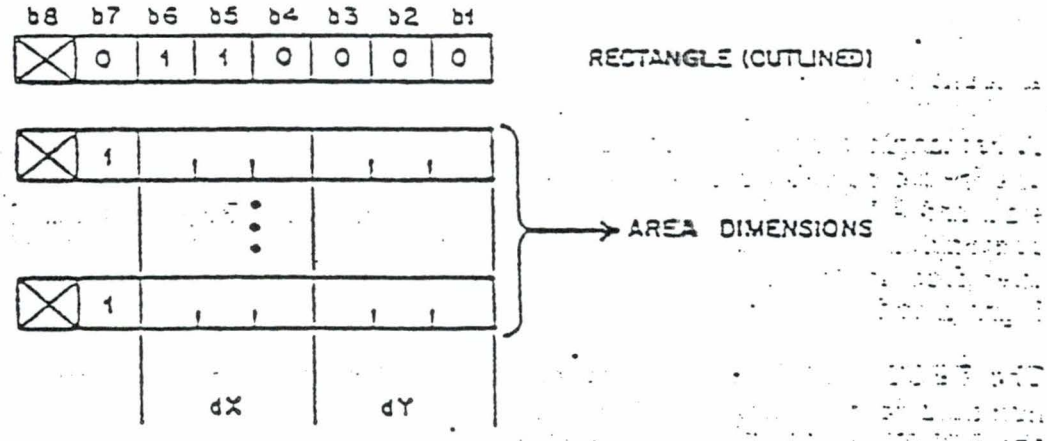


Figure 45

RECTANGLE (Outlined)

5.3.3.4.2.3 RECTANGLE (Filled). This opcode causes a rectangle to be drawn. The initial drawing position is the current drawing point and the width and height (dx,dy) are given as the first block or coordinate data. The rectangle is filled in the current color(s) with the current texture pattern. (See Figure 46.)

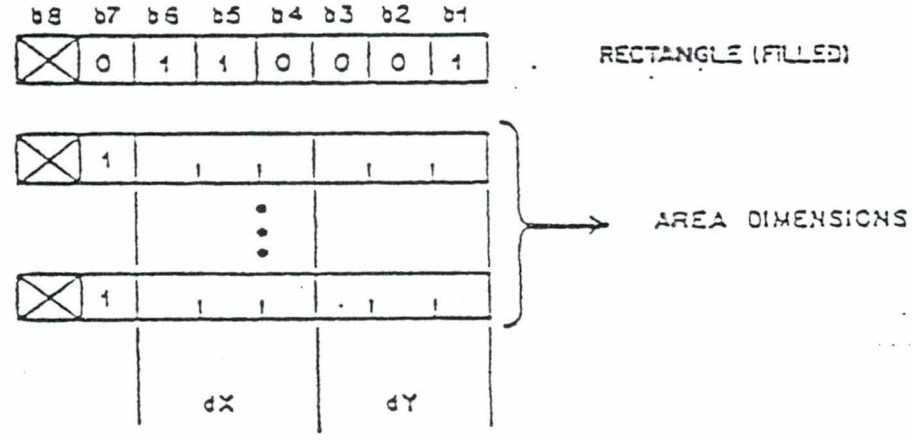


Figure 46

RECTANGLE (Filled)

5.3.3.4.2.4 SET and RECTANGLE (Outlined). This opcode causes a rectangle to be drawn. The initial drawing position is specified in absolute coordinates as the first block of coordinate data, and the width and height (dx,dy) are given as the second block of coordinate data. The rectangle is not filled. (See Figure 47.)

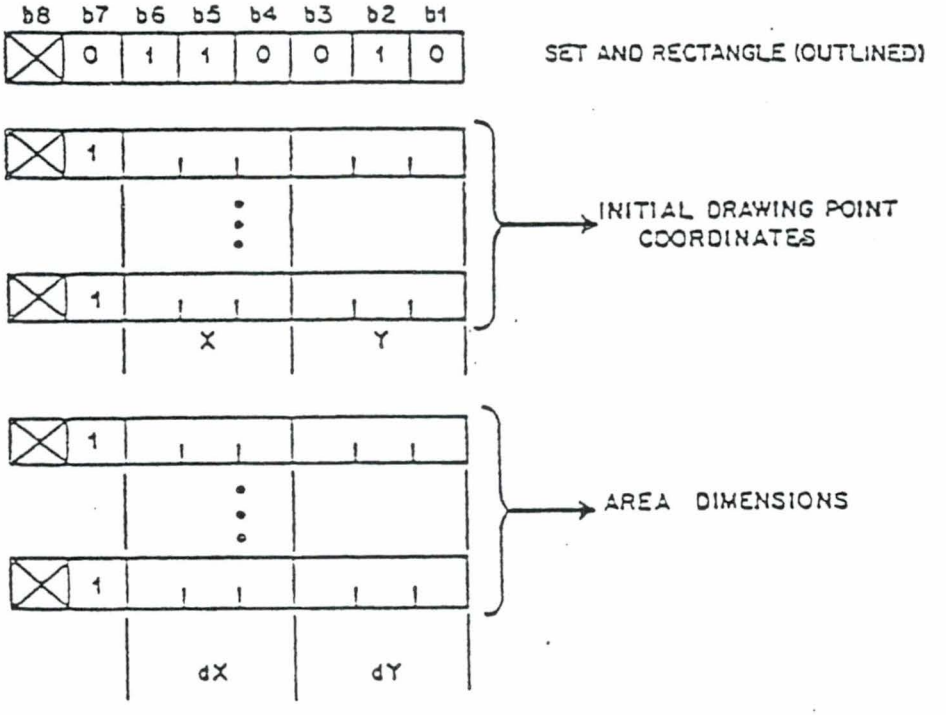


Figure 47

SET and RECTANGLE (Outlined)

5.3.3.4.2.5 SET and RECTANGLE (Filled). This opcode causes a rectangle to be drawn. The initial drawing position is specified in absolute coordinates as the first block of coordinate data, and the width and height (dx,dy) are given as the second block of coordinate data. The rectangle is filled in the current color(s) with the current texture pattern. (See Figure 48.)

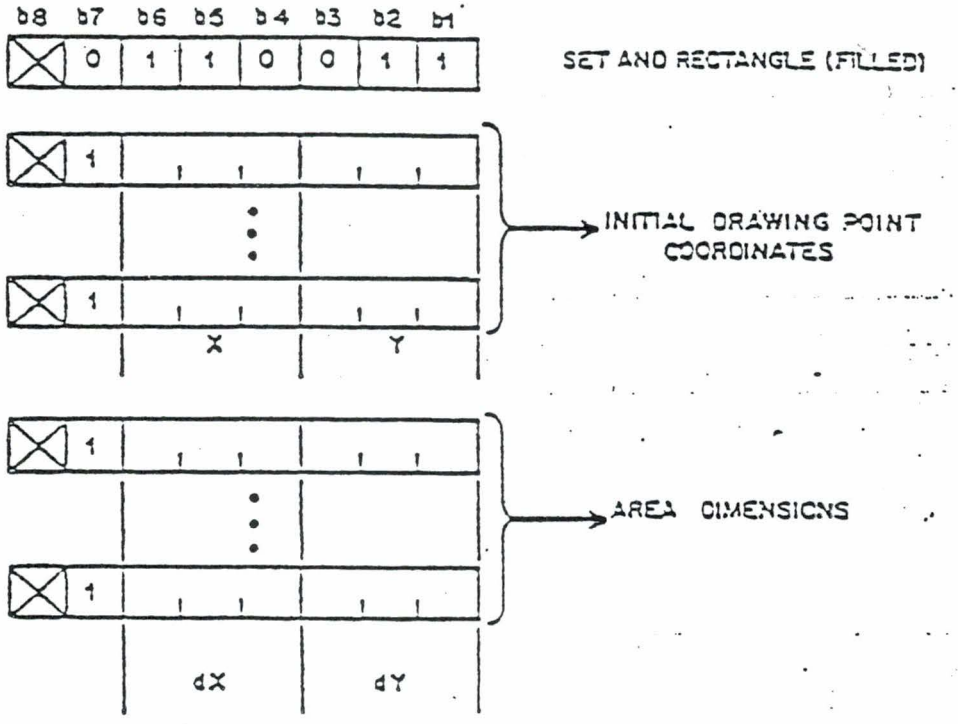


Figure 48

SET and RECTANGLE (Filled)

5.3.3.5 POLYGON

5.3.3.5.1 The POLYGON geometric drawing operation provides the capability of drawing a general polygonal area with the specified vertices. A POLYGON is specified as a series of coordinates of the vertices about the perimeter of the polygon. Each (dx,dy) coordinate pair represents a relative displacement from the last vertex (a relative displacement of magnitude 0 is ignored). There is implicit closure between the initial drawing position and the last vertex specified so that the final drawing position is identical with the initial drawing position. (See Figure 49.)

A polygon may be either filled or outlined. Outlined polygons are drawn in the in-use color(s) and have a line width that is determined by the logical pel size, and a line texture that is as specified by TEXTURE command. For filled polygons, the area enclosed is filled in the current color(s) with the texture pattern specified in TEXTURE, and the perimeter may be optionally highlighted.

A filled POLYGON must enclose a single area; that is, no line joining two consecutive vertices may cross any other line joining two consecutive vertices.

The number of vertices describing a polygon is determined by the amount of data following the POLYGON opcode. The maximum number of vertices permitted to describe a polygon is terminal dependent and a terminal must support at a minimum 256.

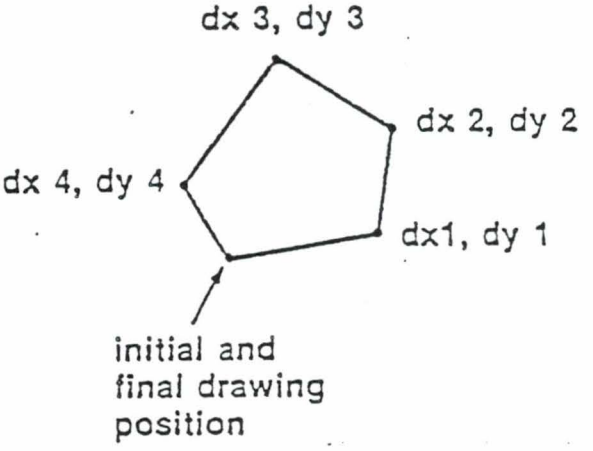


Figure 49
POLYGON

5.3.3.5.2 Commands

5.3.3.5.2.1 General. The commands used are described in Clauses 5.3.3.5.2.2 to 5.3.3.5.2.5.

5.3.3.5.2.2 POLYGON (Outlined). This opcode causes a polygon to be drawn. The initial drawing position is the current drawing point and subsequent vertex coordinates are specified as relative displacements from the previous vertex coordinate. The polygon is not filled. (See Figure 50.)

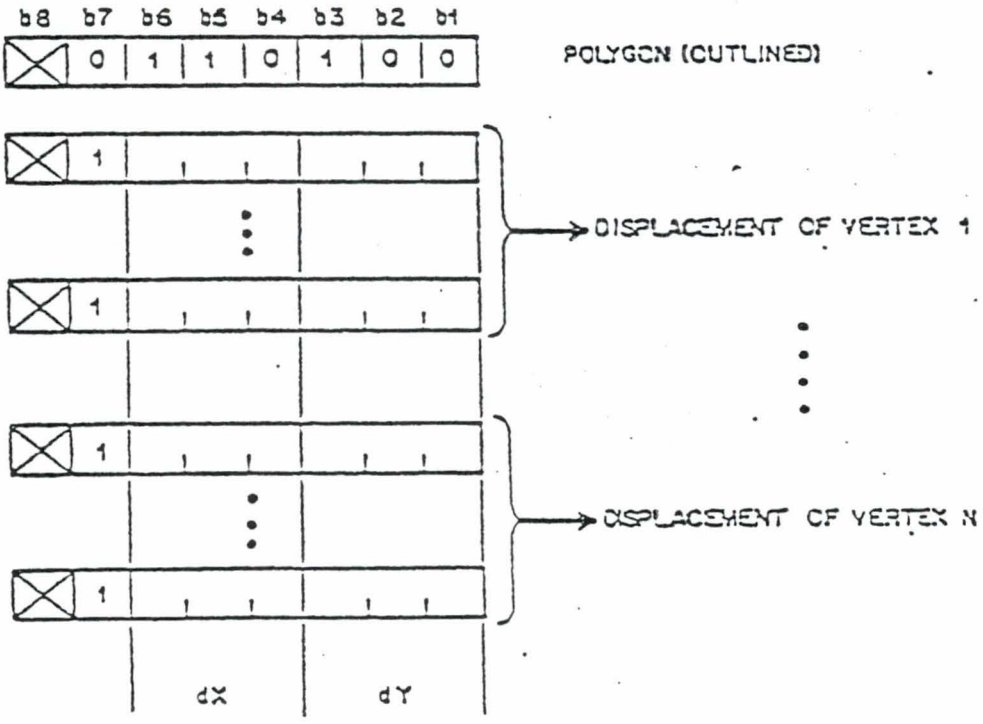


Figure 50
POLYGON (Outlined)

5.3.3.5.2.3 POLYGON (Filled). This opcode causes a polygon to be drawn. The initial drawing position is the current drawing point and subsequent vertex coordinates are specified as relative displacements from the previous vertex coordinate. The polygon is filled in the current color(s) with the current texture pattern. (See Figure 51.)

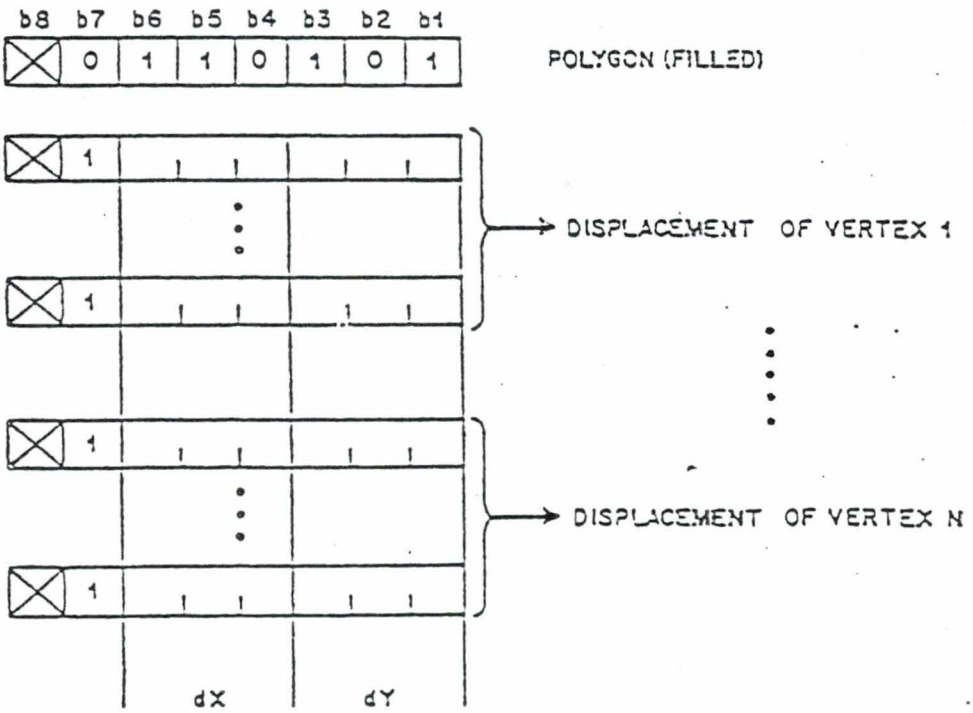


Figure 51
POLYGON (Filled)

5.3.3.5.2.4 SET and POLYGON (Outlined). This opcode causes a polygon to be drawn. The initial drawing position is specified in absolute coordinates as the first block of coordinate data, and subsequent vertex coordinates are specified as relative displacements from the previous vertex coordinates. The polygon is not filled. (See Figure 52.)

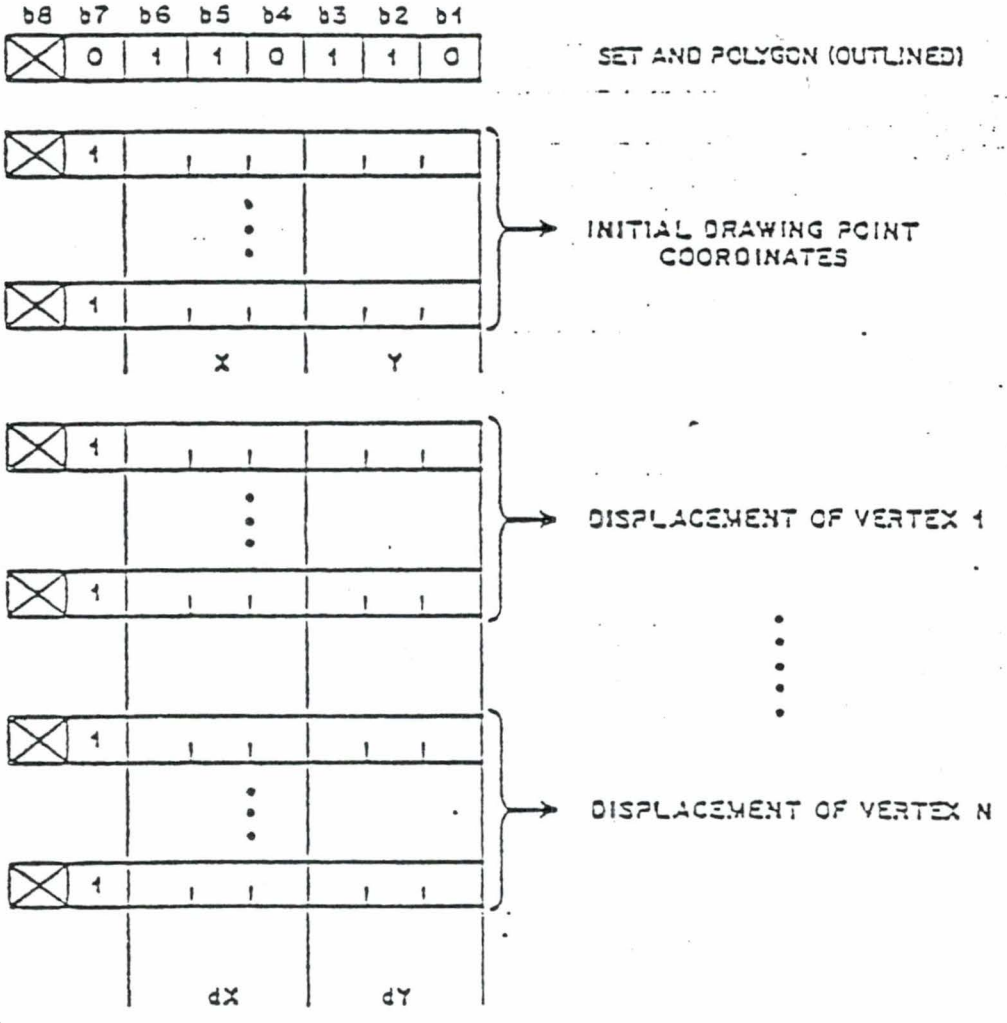


Figure 52

SET and POLYGON (Outlined)

5.3.3.5.2.5 SET and POLYGON (Filled). This opcode causes a polygon to be drawn. The initial drawing position is specified in absolute coordinates as the first block of coordinate data, and subsequent vertex coordinates are specified as relative displacements from the previous vertex coordinates. The polygon is filled in the current color(s) with the current texture pattern. (See Figure 53.)

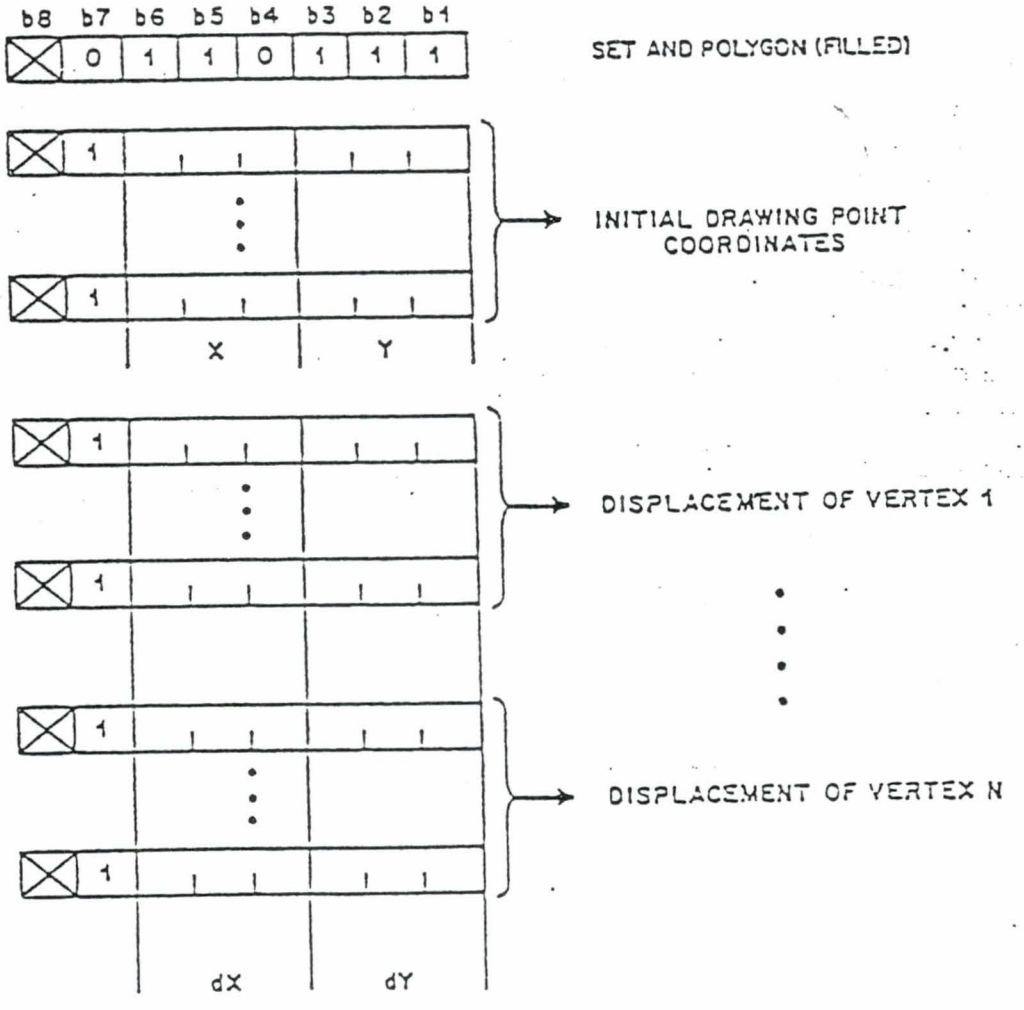


Figure 53

SET and POLYGON (Filled)

5.3.3.6 INCREMENTAL

5.3.3.6.1 The INCREMENTAL opcodes allow for the specification of complex images in a compact manner. There are four INCREMENTAL opcodes, namely FIELD, INCREMENTAL POINT, INCREMENTAL LINE, and INCREMENTAL POLYGON (Filled).

The image may be of a photographic nature (FIELD and INCREMENTAL POINT) or may consist of complex lines such as signatures (INCREMENTAL LINE) or filled polygons such as logos or other symbols (INCREMENTAL POLYGON).

5.3.3.6.2 FIELD. This opcode is used to define the active ~~drawing area~~ used by INCREMENTAL POINT (the active ~~drawing area~~ is also used for columnated text and for unprotected fields). The origin point of the field is specified in absolute coordinates (X,Y) as the first block of coordinate data. The next block of coordinate data gives the field dimensions, width and height (dx,dy). Note that dx and/or dy may be positive or negative, so that the origin point may be placed in any of the four corners of the ~~drawing area~~. The current drawing point is set to the origin of the field after FIELD has been executed. Only one active ~~drawing area~~ may be defined at a time: i.e., execution of FIELD will undefine any previous active ~~drawing area~~. If no data bytes follow the FIELD opcode, the active ~~drawing area~~ is set to the full unit screen (see Figure 54). The default active ~~drawing area~~ is the unit screen.

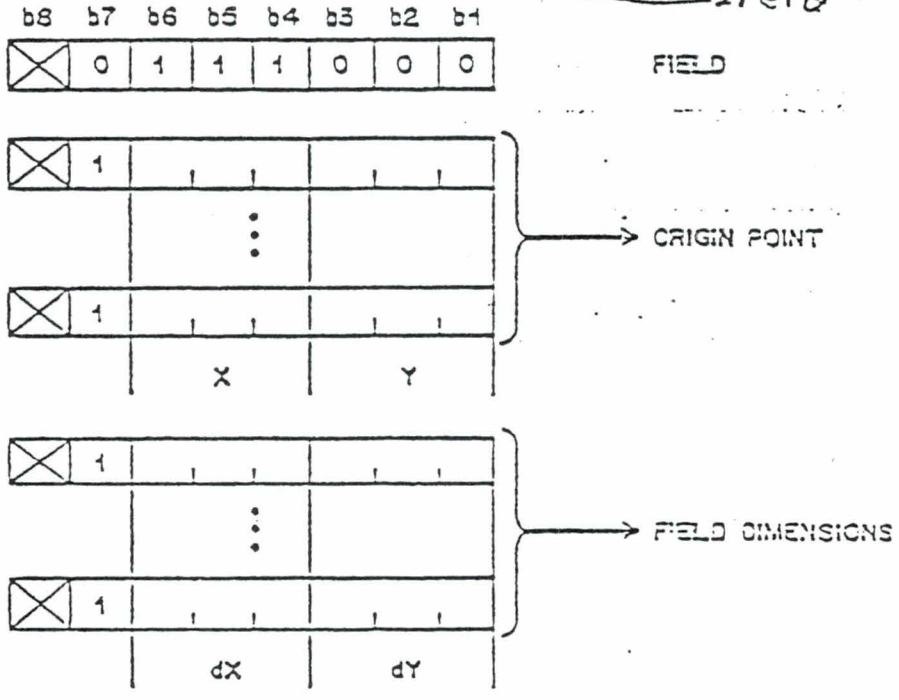


Figure 54
FIELD

5.3.3.6.3 INCREMENTAL POINT. With the INCREMENTAL POINT command, an image can be described as a string of color specifications that are deposited in a raster-sequential manner within the active ~~drawing area~~ field. These color specifications are contained in a string operand. In color mode 0, these will be interpreted as actual color values. In color modes 1 and 2, these will be interpreted as color map addresses. The algorithm by which the image is constructed is as follows:

Step 1 The operation starts at the current drawing point.

Step 2 If no portion of the logical pel exceeds the active ~~drawing area~~ field, then the first color obtained from the string operand is deposited in the display memory location(s) corresponding to the pixel(s) lying under the logical pel. If a pixel lies under the logical pel associated with the drawing point for more than one deposit operation, it retains the color deposited there last. The drawing point is then automatically moved in the X direction a distance equal to the width (dx) of the logical pel. Note that if dx is positive, the drawing point moves to the right and if dx is negative, the drawing point moves to the left. The next color is then obtained and the process is repeated.

Step 3 If any portion of the logical pel exceeds the active ~~drawing area~~ field, then any remaining bits in the byte of the string operand currently being interpreted are discarded, even if there is a sufficient number of bits remaining to make up a complete color specification. Interpretation resumes at the first bit (i.e., b6) in the next complete byte. If there are no further numeric data bytes, then the operation is terminated; otherwise the drawing point is repositioned to the opposite boundary. If moving the drawing point in the Y direction a distance equal to the height (dy) of the logical pel would cause any portion of the logical pel to exceed the active field ~~drawing area~~, then the Y value is left constant and the entire display image lying within the area of the screen defined by the active field ~~drawing area~~ is scrolled in the opposite direction, i.e., a distance equivalent to -dy; otherwise the drawing point is moved in the Y direction a distance equal to the height (dy) of the logical pel. Note that if dy is positive, the drawing point moves up and if dy is negative, the drawing point moves down. If the operation has not terminated, then the process continues at Step 2. If the operation has terminated, the drawing point is set to the origin of the active field.

The INCREMENTAL POINT opcode and its operands are shown in Figure 55. An example of an INCREMENTAL POINT picture is shown in Figure 56.

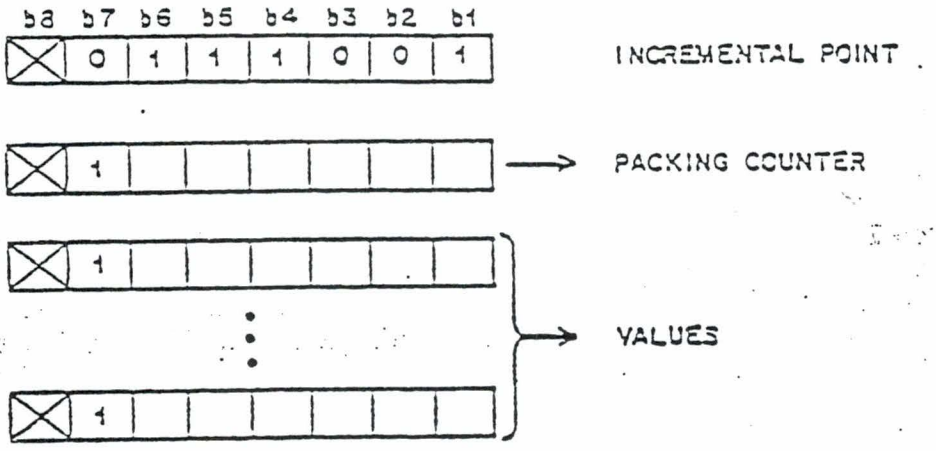


Figure 55

Figure 56

The INCREMENTAL POINT command takes two operands. The first is a single byte, fixed format operand that describes the packing counter. The packing counter is an unsigned integer that determines the number of consecutive bits to be taken from the string operand to make up a single color specification. (A packing counter of 0 is, by convention, taken to indicate a count of 64.) The second operand is a string operand of indeterminate length. It contains the color specifications, stored sequentially without regard to byte boundaries, high order bit (b6) to low order bit (b1) within the numeric data fields. In color mode 0 these color specifications are interpreted as actual color values; that is, a number of bits equal to the packing count is used to define the color applied to each drawing operation. As in the multi-value color specification, the bits are organized into three-tuples and the order of interpretation of the bits is G, R, B. Note that a single color specification may contain multiple three-tuples depending on the packing count, in which case the first three-tuple contains the MSBs and the last contains the LSBs of the three primaries. For example, if the packing count were 6, the color specification for each drawing operation would look like GRBGRB and each primary would be specified to two bits of accuracy. If the packing count is not an integer multiple of three, then each primary will not be specified to an equal accuracy. For example, if the packing count were 4, the color specification for each drawing operation would look like GRBG. Note that these would be concatenated within the string operand without regard to byte boundaries. In this example, then, the bits in the string operand would look like GRBGGRBG....

In color modes 1 and 2, these color specifications are interpreted as ordinal numbers, i.e., as addresses into a previously loaded color map. Again, a number of bits equal to the packing count is used to define the color applied to each drawing operation. These specifications are concatenated in the string operand without regard to byte boundaries.

It ~~may be desirable~~ is required, depending on the relationship between the logical pel dimensions and the physical pixel dimensions on an individual display device, to perform a pre-execution rescaling of both the logical pel dimensions and the active ~~drawing area~~ field dimensions to avoid certain types of distortions that will result from a mismatch. The goal of this type of rescaling ~~would be~~ is to make the logical pel dimensions become either integer multiples or integer fractions of the corresponding physical pixel dimensions. (The active ~~drawing area~~ field dimensions would have to be scaled equivalently in order ~~to maintain line synchronism~~ in the image.) ~~This type of terminal dependent implementation is~~ permissible as long as 1) the logical pel dimensions and active ~~drawing area~~ field dimensions are restored to their pre-scaled values after the INCREMENTAL POINT command completes execution, and 2) the resultant image is guaranteed to lie within the original active ~~drawing area~~ field, and 3) the implementation ensures that shear does not occur for all possible precisions of the logical pel and field dimensions as specified by the domain command.

If INCREMENTAL POINT is received and ~~there is an active drawing area defined but~~ the initial drawing point lies outside ~~that area,~~ the active field, the command is considered to be in error and is rejected in its entirety; i.e., it is executed as a null operation. If INCREMENTAL POINT is received and either or both dimensions of the logical pel are equal to 0, then these dimensions are set, for the duration of the command only, to the smallest positive value specifiable within the current domain. (For example, if the current multi-value operand length is 3 bytes, then the smallest specifiable value would be +.00000001, i.e., 1/256.)

5.3.3.6.4 INCREMENTAL LINE. The INCREMENTAL LINE geometric drawing operation provides the capability of compactly describing an image consisting of a series of short line segments drawn in the current color and line texture. (See Figures 57 and 58.)

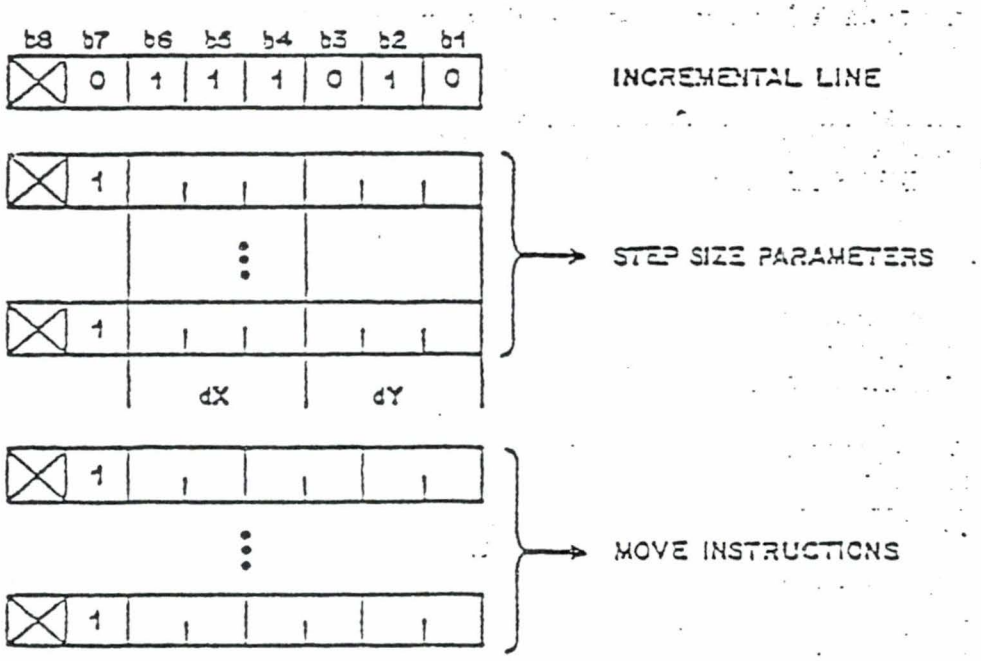


Figure 57

INCREMENTAL LINE

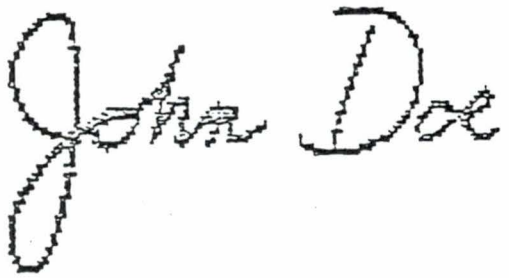


Figure 58

Example INCREMENTAL LINE Picture

The first multi-value operand specifies the step size parameters, dx and dy, as signed displacements.

The last block of data is a string operand that gives the move values as an indefinite number of bytes, each of which contains three two-bit nibbles in the numeric data field that are interpreted b6 to b1. The interpretation of these two-bit nibbles is as shown in Table 18.

Table 18

Incremental Line Move Values

Nibble Value		Interpretation
0	0	Interpret the following nibble as a "modify parameter" instruction (see Table 19).
0	1	Advance the drawing point a distance dx in the x direction and optionally draw a line.
1	0	Advance the drawing point a distance dy in the y direction and optionally draw a line.
1	1	Advance the drawing point a distance dx in the x direction and dy in the y direction and optionally draw a line.

If the draw flag is on, then every time the drawing point is stepped, a line in the current line texture and color is drawn joining the current drawing point (after the step operation) with the previous drawing point (before the step operation). If the draw flag is off, then no line is drawn after the step operation. When a nibble value of (0,0) is encountered (Table 18), the next nibble is interpreted as a "modify parameter" instruction as shown in Table 19. The nibble following that is interpreted as a step operation (Table 18). Note that the draw flag is initially on whenever the INCREMENTAL LINE opcode is encountered. When the string operand is terminated, the current drawing point is left at the final drawing point.

Table 19

Incremental Line Modify Parameter Instructions

Nibble Value		Modify Parameter Instruction
0	0	Change the state of the draw flag (on to off if originally on, or off to on if originally off).
0	1	Change sign of dx.
1	0	Change sign of dy.
1	1	Change sign of dx and dy.

5.3.3.6.5 INCREMENTAL POLYGON (Filled). The INCREMENTAL POLYGON geometric drawing operation provides the capability of compactly describing a polygon drawn with a series of short line segments and filled in the current color and texture pattern. The highlight attribute also applies to this PDI. (See Figures 59 and 60.)

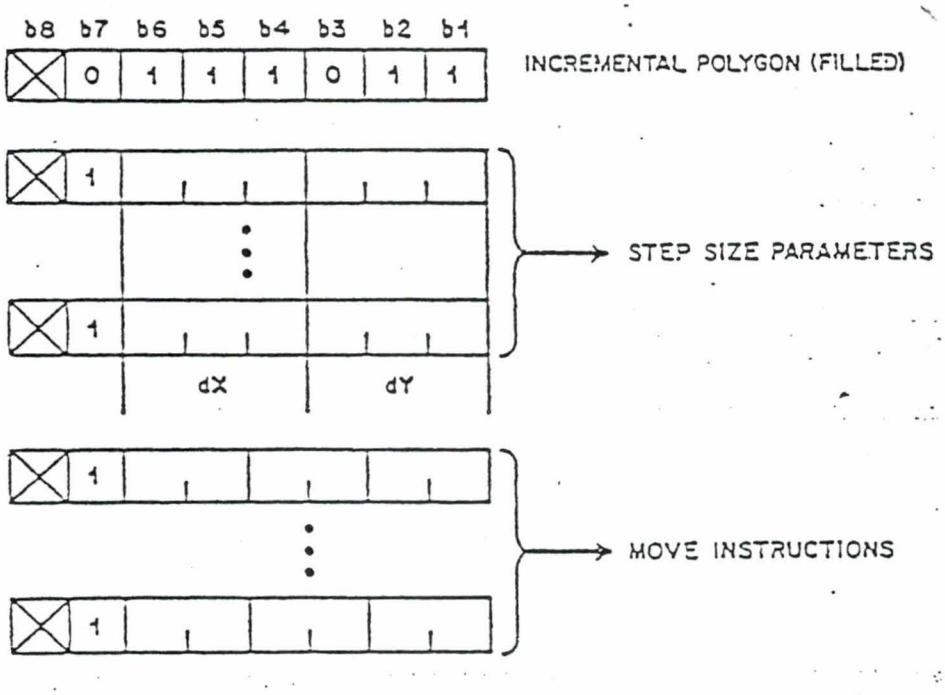


Figure 59

INCREMENTAL POLYGON (Filled)

Figure 60

Example INCREMENTAL POLYGON PICTURE

The interpretation of the operand data following the opcode is exactly as for the INCREMENTAL LINE opcode, with the following exceptions:

(a) The draw flag is always on;

(b) Should a "modify parameter" instruction (Table 19) to change the state of the draw flag (0,0) be encountered, it is treated as a null, and the following nibble is interpreted as the "modify parameter" instruction (Table 19);

(c) The final drawing point is implicitly taken as the initial drawing point;

(d) The resulting figure is filled in the current color and texture pattern and according to the highlight attribute.

Note that INCREMENTAL POLYGON must enclose a single area (similar to POLYGON, Clause 5.3.3.5).

5.4 Mosaic Set. Figure 62 shows the character assignments for the Mosaic Set. This comprises sixty-five 2 x 3 block mosaic characters including a second copy of the solid mosaic in position 5/15. The remaining character positions are not to be used. Mosaic characters can be displayed in two modes, contiguous or separated, depending on the underline mode described in Clause 6.2.7.15. In contiguous mode, the six mosaic elements that compose each character should completely span the given character field at any size. In separated mode, each of the mosaic elements that are illuminated are reduced in the horizontal dimension by the width (dx) of the logical pel size, and are reduced in the vertical dimension by the height (dy) of the logical pel size. (The logical pel is described in Clause 5.3.2.2.) The reduced mosaic elements, in this case, are left and bottom justified within the normal element area, the remainder of the area being considered as background. The graphic symbol in position 2/0 of the Mosaic Set is not subject to underline. Mosaics (in separated or contiguous mode) are not subject to proportional spacing.

The algorithm that determines which of the sub-elements is on is derived from the bit combinations of the code table reference. (See Figure 61.)

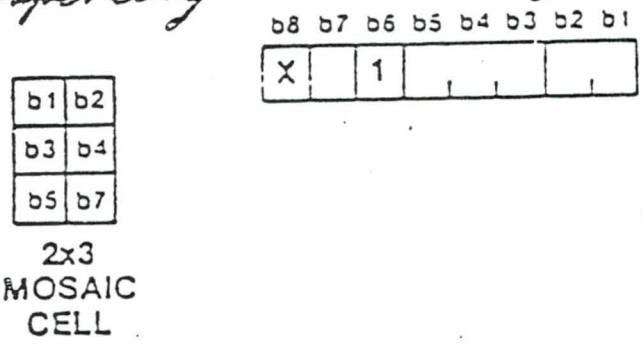


Figure 61

Mosaic Sub-element Encoding

The code combination 5/15 (1011111) also implies that all sub-elements are on. See Figure 62 for a code table representation of the mosaic scheme. The F character designation for the Mosaic Set is 7/13.

Figure 62

Mosaic Set

5.5 Macro Set. The macro feature provides the capability of encoding sequences of presentation level codes to be executed upon command. A macro definition consists of an arbitrary string of locally buffered presentation level code that is identified by a code from the macro G-set. This name thereafter acts as a substitute for the entire string of characters that make up that particular macro. Up to 96 macros can be defined simultaneously (see Clause 6.2.2). A macro can be used by designating the macro set as one of the G-sets, followed by invoking the macro set into the in-use table and transmitting the macro code. A macro code may also be included within any macro definition, thereby providing a nesting capability. It is essential that the application level assume responsibility for infinite loops. Any macro may be linked to a function key on the keyboard to allow its execution to be activated by the user.

5.6 Dynamically Redefinable Character Set (DRCS). Unlike the other character sets, whose pattern definitions are permanently stored in the terminal and cannot be altered by the host computer, the Dynamically Redefinable Character Set (DRCS) designated by a three character escape sequence provides a facility whereby a maximum of 96 custom defined patterns can be downloaded and utilized in an ~~identical~~ similar manner as the Primary Supplementary, ~~and Mosaic sets~~. At display time, ~~therefore~~, they are subject to the same attributes. The manner in which the patterns are actually downloaded is described in Clause 6.2.3.

as alpha-numeric text.

6. Coding of C Sets

6.1 C0 Control Set

6.1.1 This clause describes the C0 control set (see Figure 63) that occupies columns 0 and 1 of the 7- and 8-bit in-use tables. The functions are as follows.

6.1.2 Format Effector Characters

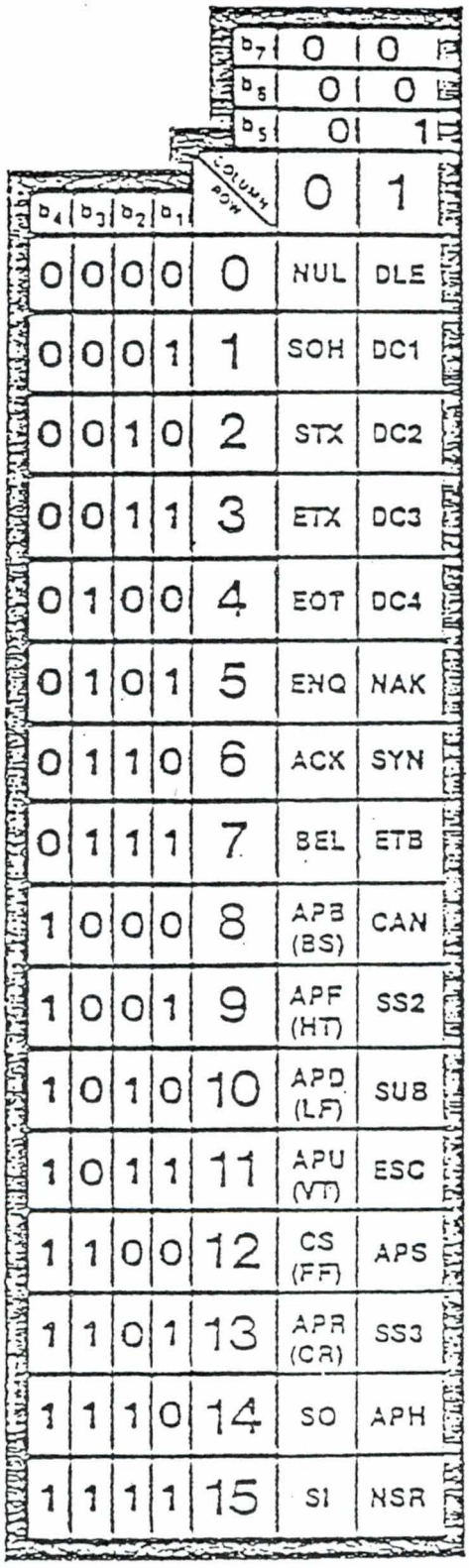
6.1.2.1 APB—ACTIVE POSITION BACKWARD (0/8) (backspace) is used to position the cursor a distance equal to the inter-character spacing lying parallel to the character path in the direction opposite to the character path (i.e., 180° from the direction of the character path). If such a movement would cause the edge of the unit screen (or active ~~drawing area~~ should the cursor lie within such ~~an area~~) to be crossed, then the cursor is instead positioned at the opposite edge (along the character path) of the screen or active ~~drawing area~~ and an automatic APU is executed.

6.1.2.2 APF—ACTIVE POSITION FORWARD (0/9) (horizontal tab) is used to position the cursor a distance equal to the inter-character spacing lying parallel to the character path in the direction of the character path. If such a movement would cause the edge of the unit screen (or active ~~drawing area~~ should the cursor lie within such ~~an area~~) to be crossed, then the cursor is instead positioned at the opposite edge (along the character path) of the screen or active ~~drawing area~~ and an automatic APD is executed.

6.1.2.3 APD—ACTIVE POSITION DOWN (0/10) (line feed) is used to position the cursor a distance equal to the inter-row space lying perpendicular to the character path in a direction perpendicular to the character path (-90°). If such a movement would cause the edge of the unit screen (or active ~~drawing area~~ should the cursor lie within such ~~an area~~) to be crossed, then special action is taken that is dependent on whether or not scroll mode is in effect (see Clauses 6.2.7.13 and 6.2.7.14).

6.1.2.4 APS—ACTIVE POSITION SET (1/12) is used to set the cursor position without resetting any parameters or attributes.

The two bytes immediately following an APS shall both come from columns 2 through 7 of the in-use table. They represent the row address and column address, respectively, to which the cursor is to be moved. The row address is obtained from the first byte following an APS by taking the binary integer comprising bits b7 through b1 with b7 being the MSB, and subtracting 32. Similarly, the column address is obtained from the second byte following an APS by taking the binary integer comprising bits b7 through b1 with b7 being the MSB, and subtracting 32. This gives an address range from 0 through 95 inclusive for the row and column addresses. For example, the bit combination 3/6 yields the binary integer 54, which, after subtracting 32, gives the address 22.



b_7	0	0	0	0	0	0	0	0
b_6	0	0	0	0	1	1	1	1
b_5	0	0	1	1	0	0	1	1
ROW / COLUMN	0	1	0	1	0	1	0	1
b_4 b_3 b_2 b_1								
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	APB (BS)	CAN	(	8	H	X	h	x
1 0 0 1	9	APF (HT)	SS2	)	9	I	Y	i	y
1 0 1 0	10	APD (LF)	SUB	*	:	J	Z	j	z
1 0 1 1	11	APU (VT)	ESC	+	;	K	[	k	{
1 1 0 0	12	CS (FF)	APS	,	<	L	\	l	|
1 1 0 1	13	APR (CR)	SS3	-	=	M	]	m	}
1 1 1 0	14	SO	APH	.	>	N	^	n	~
1 1 1 1	15	SI	NSR	/	?	O	_	o	DEL

Figure 63

C0 Control Set

(with the default inter-character and inter-row spacing). The cursor is positioned assuming zero character rotation to establish the character field origin. Once the character field origin is established, the character field and cursor are rotated, if necessary.

Rows and columns are numbered starting with row 0, column 0, in the lower leftmost character position of the physical display area, and refer to the nominal screen format established by the ~~in-use~~ current character field size.

6.1.2.5 APU—ACTIVE POSITION UP (0/11) (vertical tab) is used to position the cursor a distance equal to the inter-row space lying perpendicular to the character path in a direction perpendicular to the character path (90°). If such a movement would cause the edge of the unit screen (or active ~~drawing area~~ field should the cursor lie within such ~~an area~~ a field) to be crossed, then special action is taken that is dependent on whether or not scroll mode is in effect (see Clauses 6.2.7.13 and 6.2.7.14).

6.1.2.6 CS—CLEAR SCREEN (0/12) is used to position the cursor to the upper left character position of the physical display area. In color mode 0, it clears the physical display area to nominal black. ~~In color mode 1, it clears the physical display area to the color that is at color map address zero (nominal black).~~ In color mode 2, it clears the physical display area to the current in-use background color.

6.1.2.7 APR—ACTIVE POSITION RETURN (0/13) (carriage return) is used to position the cursor to the first character position within the physical display area (or within the active ~~drawing area~~ field should the cursor lie within such an ~~area~~ field) along the character path.

6.1.2.8 APH—ACTIVE POSITION HOME (1/14) is used to position the cursor to the upper left character position in the physical display area.

6.1.3 Code Extension Control Characters

6.1.3.1 SO—SHIFT OUT (0/14) is used to invoke the G1 set into the in-use table (see Clauses 4.3.2 and 4.3.3).

6.1.3.2 SI—SHIFT IN (0/15) is used to invoke the G0 set into the in-use table (see Clauses 4.3.2 and 4.3.3).

6.1.3.3 SS2—SINGLE SHIFT TWO (1/9) is used to invoke the G2 set into the in-use table in a non-locking manner (see Clauses 4.3.2 and 4.3.3).

6.1.3.4 SS3—SINGLE SHIFT THREE (1/13) is used to invoke the G3 set into the in-use table in a non-locking manner (see Clauses 4.3.2 and 4.3.3).

6.1.3.5 ESC—ESCAPE (1/11) is used for code extension. See Clauses 4.3.2 and 4.3.3.

6.1.4 Transmission Control Characters—The transmission control characters, i.e., SOH (0/1), STX (0/2), ETX (0/3), EOT (0/4), ENQ (0/5), ACK (0/6), DLE (1/0), NAK (1/5), SYN (1/6), and ETB (1/7), are reserved for use at protocol levels other than the presentation level.

6.1.5 Device Control Characters—The device control characters, i.e., DC1(1/1), DC2(1/2), DC3(1/3), DC4(1/4), are reserved for use at protocol levels lower than the presentation level.

6.1.6 Other Control Characters

6.1.6.1 NUL—NULL (0/0) has no effect on the presentation level.

6.1.6.2 BEL—BELL (0/7) is used to momentarily ring a bell or effect another transient indication.

6.1.6.3 CAN—CANCEL (1/8) is used to terminate processing of all currently executing macros. Execution is resumed at the next presentation level character following the terminated macro call. The effect of CAN is immediate, i.e., it is not put at the end of any existing queue of unprocessed presentation level code.

6.1.6.4 SUB—The precise meaning of SUB (1/10) is reserved for future standardization and is treated as a null operation.

6.1.6.5 NSR—NON-SELECTIVE RESET (1/15) serves two functions; it non-selectively resets the presentation process as defined below and it can be used as an alternative means to position the cursor. When an NSR is received, the following action is taken:

(a) The four active G-sets and the in-use coding table are set to their default states, as described in Clause 4.

(b) The DOMAIN parameters are set to their default values, as described in Clause 5.3.2.2;

(c) The text parameters (from the TEXT opcode, from the C1 Set and the ~~Active Drawing Area~~ active field), as described in Clause 5.3.2.9.2.2, are set to their default values;

(d) The TEXTURE parameters are set to their default values, as described in Clause 5.3.2.4. Note that the programmable masks are not cleared;

(e) The color mode is set to color mode 0 and the in-use drawing color is set to white. (Note that the color map, ~~if any,~~ is not affected.)

The cursor is positioned using the two bytes immediately following the NSR. If the two bytes are both from columns 4 through 7 of the in-use table, then they are taken to represent, in binary form (i.e., the binary digit comprising the bits b6 through b1 with b6 being the MSB), the row and column address, respectively, to which the cursor should be moved. Rows and columns are numbered starting with row 0, column 0 in the upper leftmost character position on the physical display area and refer to the nominal screen format established by the default character size. If the two bytes following the NSR are not both from columns 4 through 7 of the in-use table, they are both rejected (i.e., treated as a null operation) and the cursor is not repositioned.

6.2 C1 Control Set

6.2.1 The C1 control set (see Figure 64) is used to allow control over text format and also to create macros, DRCS, programmable texture masks, and unprotected fields. These capabilities are described in Clauses 6.2.2 to 6.2.6.

b_4	b_3	b_2	b_1	COLUMN / ROW	A	B
0	0	0	0	0	DEF MACRO @	PROTECT P
0	0	0	1	1	DEF MACRO A	EDC1 Q
0	0	1	0	2	DEF MACRO B	EDC2 R
0	0	1	1	3	DEF GRCS C	EDC3 S
0	1	0	0	4	DEF TEXTURE D	EDC4 T
0	1	0	1	5	END E	WORD WRAP ON U
0	1	1	0	6	REPEAT F	WORD WRAP OFF V
0	1	1	1	7	REPEAT TO EOL G	SCROLL ON W
1	0	0	0	8	REVERSE VIDEO H	SCROLL OFF X
1	0	0	1	9	NORMAL VIDEO I	UNDER LINE START Y
1	0	1	0	10	SMALL TEXT J	UNDER LINE STOP Z
1	0	1	1	11	MED TEXT K	FLASH CURSOR [
1	1	0	0	12	NORMAL TEXT L	STEADY CURSOR \
1	1	0	1	13	DOUBLE HEIGHT M	CURSOR OFF]
1	1	1	0	14	BLINK START N	BLINK STOP ^
1	1	1	1	15	DOUBLE SIZE O	HARD TEST _

Definition of Columns A and B

(1) If a C1 control function is represented by a 2-character escape sequence (in a 7-bit code), the table specifies the bit combination of the final character by taking A = 4 and B = 5.

(2) If a C1 control function is represented by a single 8-bit combination, the table specifies this combination by taking A = 8 and B = 9.

Figure 64

C1 Control Set

6.2.2 Macros: General

6.2.2.1 DEF MACRO. This C1 control is used to define a macro. The character following this control is interpreted as the name of the macro; e.g., the character (2/0) would cause macro M0 to be defined. All characters subsequent to this character are stored (but not executed) within the terminal under the specified macro name. Definition of the macro terminates upon receipt of one of the following C1 controls: DEF MACRO, DEFP MACRO, DEFT MACRO, DEF DRCS, or END. Neither the terminating control character nor its preceding ESC character in a 7-bit environment is stored as part of the macro.

Definition of a macro replaces any previously existing macro under the same name. A null macro definition, i.e., a macro definition in which there are no characters between the macro name and the terminating C1 character (or its preceding ESC character in a 7-bit environment) causes that macro to be deleted. Definition of a macro is independent of whether the macro set is invoked or not (though it must, of course, be invoked in order to actually execute the macro).

All macros may be simultaneously deleted with the RESET command.

6.2.2.2 DEFP MACRO. The operation of this control character is identical to that of the DEF MACRO command, except that:

(a) Incoming characters that make up the macro definition are simultaneously executed and stored;

(b) DEFP MACRO cannot be used to define a macro that calls itself.

6.2.2.3 DEFT MACRO. The DEFT MACRO control character is used to define a transmit-macro. Transmit-macros, when called, are not executed, but are transmitted in their entirety to the host computer or to a local application process. This could, for example, be used to provide a programmable-function key capability if one or more keys on the keyboard were capable of causing the execution of macros.

Transmit-macros are defined and deleted in a manner similar to that described for normal macros, and they share the same 96 macro names.

6.2.3 DEF DRCS. The DEF DRCS command is used to start the downloading operation for one of the DRCS characters, of which a total of 96 are permitted. This is followed by the bit combination representing the DRCS character being defined. Next comes any valid stream of presentation level code. The DRCS downloading operation is terminated when an END, DEF MACRO, DEFP MACRO, DEFT MACRO, DEF TEXTURE, or another DEF DRCS command is received. When the current DRCS downloading operation is

A macro may be longer or shorter than the previously existing macro which it replaces.

terminated by another DEF DRCS command, the next character of the DRCS G-set (i.e., in the sequence 2/0, 2/1, . . . 2/15, 3/0, 3/1 . . . .) is defined by the presentation-level code immediately following this new DEF DRCS command. (This is the only time DEF DRCS is not followed by the DRCS character to be defined.) If a DRCS definition is immediately terminated with no intervening presentation level code, the buffer space allocated to that character is freed.

Definition of a DRCS set is independent of whether the set is invoked or not (though it must, of course, be accessed from the in-use table in order to actually display the character).

The presentation level code defining the DRCS character is executed within the unit screen upon being received by the terminal. It is not, however, displayed ~~on the screen,~~ but rather is used to modify a separate storage buffer that is mapped from the unit screen. The storage buffer is one bit deep (i.e., on or off only), has a width ~~equal~~ to the current character width ~~(in pixels), and a height equal to the current character height (in pixels).~~ This buffer will be used in a manner similar to the permanently stored primary character set patterns and will be subject to the same attributes (character size, in-use color, etc.). The character field ~~is used to define~~ the aspect ratio and coarseness of the character definition. For example, a character that is 6 ~~units~~ wide and 10 ~~units~~ high would be defined on a coordinate system of 0.0 to 0.6 in the X axis and 0.0 to 1 in the Y axis.

The buffer is initially cleared to all 0s upon receipt of the DRCS character (or another DEF DRCS command when the next DRCS character in sequence is implied). All presentation level code is then executed, causing the appropriate bits in the DRCS buffer to be set to ones to define the DRCS shape. SET COLOR, SELECT COLOR, and BLINK will be executed, but will have no effect on the DRCS character definition. If the text size is changed within the DRCS definition, it will have an effect upon the generation of subsequent text characters within the unit screen and their mapping into the DRCS buffer, but it will not affect the size of the DRCS buffer, since it has already been allocated.

After the downloading sequence has been terminated, the terminal reverts to the normal procedure of mapping the unit screen to the display screen with the drawing point reset to (0,0).

The entire DRCS character set may be cleared using the RESET command. Should a RESET that clears the DRCS be received during a downloading operation, it will clear the character pattern definition in progress, but the downloading operation will continue. The effect on the screen of the redefinition or deletion of a DRCS character that is being displayed is undefined.

When using INCREMENTAL POINT within a DRCS definition, the packing counter operand shall be set to one by the sending presentation process. The portion of the unit screen on which the DRCS character is defined shall have an aspect ratio equal to the aspect ratio of the current character field, with the greater dimension set to unity.

except that the texture mask size is used rather than the character field size.

6.2.4 DEF TEXTURE. This command is used to define one of the four programmable texture masks described in Clause 5.3.2.4.

The DEF TEXTURE character is immediately followed by one of the following bit combinations: (4/1), (4/2), (4/3), (4/4), which causes selection of mask A, B, C, or D respectively to be defined. Any existing texture pattern associated with the specified mask is deleted. The mask is cleared by terminating the command at this point. If presentation-level code follows, it describes the texture mask in the same manner as DRCS characters. All of the special restrictions that apply to the downloading of DRCS patterns also apply to the downloading of texture masks. At the end of the downloading sequence, the terminal reverts to the normal procedure of mapping the unit screen to the display screen with the drawing point reset to (0,0).

6.2.5 END. This command terminates the current DEF MACRO, DEFP MACRO, DEFT MACRO, DEF DRCS, or DEF TEXTURE operation. It is also used in the transmission of data in an unprotected field to the host (see Clause 6.2.6).

6.2.6 Protect/Unprotect. Unprotected fields are rectangular areas on the display into which the user may enter or edit data for possible subsequent transmission to the host. The method of entering and editing data into these fields is left to the terminal manufacturer.

By default, the entire display screen is protected; i.e., it is not possible for the user to enter or alter data on the screen. However, if the UNPROTECT command is given, the active ~~drawing area~~ field (as defined by the FIELD command described in Clause 5.3.3.6.2) is made unprotected and the user may subsequently add or alter data within that field. Any number of unprotected fields may be defined (subject to ~~terminal~~ implementation dependent memory limitations) by defining an active ~~drawing area~~ field via FIELD command and then issuing the UNPROTECT command. Should an UNPROTECT command result in unprotecting ~~an area~~ a field that partially or wholly lies within an already unprotected field, the already unprotected field is made protected (without affecting the displayed contents) before the new ~~area~~ field is made unprotected. This prevents unprotected fields from overlapping.

The user may enter or locally edit information (text or graphics) into an unprotected field. The host may also add information into this field that may subsequently be edited by the user. When the user initiates a transmission to the host, the information entered into the unprotected field(s) is transmitted conforming to the presentation-level protocol described in this Standard. The transmit presentation process and its associated states must be maintained separately from the receive presentation process. In transmission, the FIELD command containing the coordinates of the origin of the unprotected field as well as its dimensions is transmitted, followed by the contents of the field transmitted according to the presentation level protocol defined in this Standard, and then by the END command. When more than one unprotected field is present, the order of transmission is ~~top-left to bottom-right, giving priority to Y with respect to the origin of the screen.~~ from the top to the bottom of the unit screen, using field origin points as a reference. If two or more field origins have the same Y coordinate, the order of transmission is left

Unprotected fields may be re-protected via the PROTECT command. This command causes all unprotected fields of which any portion lies within the active drawing area to be made protected. The entire screen is protected if no active drawing area is currently defined. The RESET command may be used to protect all currently defined unprotected fields.

6.2.7 Text Controls

6.2.7.1 This set of 20 controls allows simple textual functions to be implemented, some of which may also be performed via the TEXT or BLINK commands.

6.2.7.2 REPEAT. This command causes the last alphanumeric text, mosaic, or DRCS character received to be repeated a specific number of times. Bits b6 to b1 of the character following the repeat character are interpreted as the repeat count. This repeat count character must be one of the characters in columns 4 through 7 of the in-use table; otherwise the command is considered in error and is not executed.

6.2.7.3 REPEAT TO EOL. When this command is encountered, the last alphanumeric text, mosaic, or DRCS character received is repeated until the last character position along the current character path is reached. If any portion of the text cursor lies within the currently defined active drawing area when this command is encountered, then characters are repeated only up to the last character position along the current character path within the active drawing area.

6.2.7.4 REVERSE VIDEO. This command causes the terminal to enter reverse video mode in which any subsequently received alphanumeric text, mosaic, and DRCS characters are "complemented" within the current character field prior to their display. By "complemented" is meant that the pixels surrounding the character shape are turned on instead of those pixels describing the character shape itself.

6.2.7.5 NORMAL VIDEO. This command causes the terminal to exit reverse video mode. This is the default state.

6.2.7.6 SMALL TEXT. This command causes the dimensions of the character field to be set as follows:

dx = approximately 0.0125
dy = approximately 0.04

This allows an 80-column x 20-row (nominal) character display.

6.2.7.7 MEDIUM TEXT. This command causes the dimensions of the character field to be set as follows:

dx = ~~(~~approximately~~)~~ 0.03125 decimal = ~~approximately~~ 0.00001000 binary
dy = ~~(~~approximately~~)~~ 0.047 decimal = ~~approximately~~ 0.00001100 binary

~~This allows a 32 column x 20 row character display.~~

6.2.7.8 NORMAL TEXT. This command causes the dimensions of the character field to be set to their default value; that is:

dx = ~~(~~approximately~~)~~ 0.025 decimal = approximately 0.00000110 binary
dy = ~~(~~approximately~~)~~ 0.04 decimal = approximately 0.00001010 binary

~~This allows a 40 column x 20 row character display.~~

6.2.7.9 DOUBLE HEIGHT. This command causes the dimensions of the character field to be set as follows:

decimal = approximately 0.00000110 binary
dx = ~~(~~approximately~~)~~ 0.025 (default)
dy = ~~(~~approximately~~)~~ 0.08 (double default height)
decimal = approximately 0.00010100 binary

~~This allows a 40 column x 10 row character display.~~

6.2.7.10 DOUBLE SIZE. This command causes the dimensions of the character field to be set as follows:

decimal = approximately 0.00001100 binary
dx = ~~(~~approximately~~)~~ 0.05 (double default width)
dy = ~~(~~approximately~~)~~ 0.08 (double default height)
decimal = approximately 0.00010100 binary

~~This allows a 20 column x 10 row character display.~~

6.2.7.11 WORD WRAP ON. This command causes the terminal to enter word wrap mode. In this mode, subsequently received alphanumeric text is buffered into words. A word is displayed on the current line on the screen only if the entire buffered word will fit into the space remaining on the current line within the unit screen (or active ~~drawing area~~ field should the text cursor lie within that ~~area~~). If the word does not fit into the space remaining on the current line, then the cursor is repositioned beginning at the first character position on the next line within the screen or area, and the word is displayed. The SPACE character should be dropped if the last word on the line is terminated with a SPACE that does not fit on that line. For the purposes of this section, a word is defined as being an accumulation of characters between SPACE characters.

Note that SMALL TEXT, MEDIUM TEXT, NORMAL TEXT, DOUBLE HEIGHT and DOUBLE SIZE only effect the character field dimensions. They do not affect rotation, character path, inter-character and inter-row spacing, etc.

Words consisting entirely of alphabetic characters (see Table 20) and one or more embedded (i.e., not at the beginning or end of the word) special terminating characters (! " $ % ( ) [ ] < > { } ^ * + - / , . : ; = ? _ ~) may be broken between the special terminating character and the following character, which causes as much of the word to fit on the current line as possible. All other words must be kept together on a single line.

A word is also terminated by a mosaic set character, a PDI, any C-set character defined at the presentation layer except SO, SI, SS2 and SS3, or any character that causes the length of the word to equal the maximum length of a line.

6.2.7.12 WORD WRAP OFF. This command causes the terminal to exit word wrap mode. In this mode (the default mode) all text is broken on character boundaries whenever an automatic ~~carriage return~~ and ~~line feed~~ are executed.

6.2.7.13 SCROLL ON. In this mode, a subsequently received APD or APU command, which would advance the cursor out of the physical display area or active ~~drawing area~~ (if it should be in such an area), causes the entire display within the area to be scrolled. Scrolling occurs pixel-by-pixel in a direction perpendicular to the character path and far enough to bring the next intended character location just into the area.

Data scrolled out of the physical display area or active ~~drawing area~~ ~~may be buffered at the discretion of the terminal manufacturer.~~

6.2.7.14 SCROLL OFF. In this mode (the default mode) an APD or APU command, which would advance the cursor out of the physical display area or active ~~drawing area~~ (if it should be in such an area), causes the cursor to be repositioned to the opposite edge of the ~~area~~ such that the character field so defined lies entirely within the area.

6.2.7.15 UNDERLINE START. This command causes the terminal to enter underline mode. In this mode, all subsequently received primary, supplementary, ~~and~~ DRCS characters have a line added to their patterns. The line appears in the character field at the bottom and extends across the entire width of the field. Its thickness is determined by the vertical dimension (dy) of the logical pel size. The mosaic set is displayed in separated graphic mode.

6.2.7.16 UNDERLINE STOP. This command causes the terminal to exit underline mode. While in this mode (the default mode), characters from the mosaic set are displayed in contiguous mode, i.e., the six mosaic elements composing each mosaic character completely span their normal element areas.

6.2.7.17 FLASH CURSOR. This command turns on the cursor display and causes it to flash intermittently.

The color of background pixels scrolled into the field is non-black in color modes 0 and 1 and the background color in color mode 2.

6.2.7.18 STEADY CURSOR. This command turns on the cursor display and maintains it steadily visible.

6.2.7.19 CURSOR OFF. This command causes the cursor to be made invisible (default mode). Note that the cursor still functions and moves as usual; it is just not visible while in this mode.

6.2.8 OTHER CONTROLS

6.2.8.1 BLINK START. This command creates a blink process in which: the blink-from color is the current in-use drawing color; the blink-to color is black in color ~~mode 0, the color that is at color map address zero (nominal black) in color mode 1,~~ or the in-use background color in color mode 2; the on and off intervals are at the discretion of the manufacturer; and the phase delay is 0.

modes 0 and 1

nominal

In color mode 0 the C1 BLINK START command establishes an automatic process that implicitly defines a blinking color. If the in-use color is changed, the old color remains blinking and the new in-use color does not blink.

6.2.8.2 BLINK STOP. This command (the default condition) turns off any currently active blink processes utilizing the current in-use color as the blink-from color.

6.2.8.3 EDC1, EDC2, EDC3, and EDC4 (EXTENDED DEVICE CONTROLS). The precise meanings of EDC1, EDC2, EDC3, and EDC4 are reserved for future standardization, and are treated as null operations.

Note that objects drawn with the previous in-use color will remain not blinking but objects subsequently drawn with the new in-use color will blink.

7. Graphic Character Repertoire

7.1 In Tables 20 to 29 the column labelled "Coded Representation" specifies the coded representation of the graphic characters and symbols for the 7-bit and 8-bit codes. The symbol "S" is used to identify that the immediately following bit combination represents an element of the Supplementary Set. In 7-bit and 8-bit environments, the elements of the Primary Set are represented by bit combinations in the range 2/1 to 7/14 and the elements of the Supplementary Set may be represented by SS2 followed by 2/1 to 7/14 when the Supplementary Set is designated as G2 (the default). In addition, in the 8-bit case, elements of the Supplementary Set can be represented as 10/1 to 15/14 when the Supplementary Set is invoked into GR. This character repertoire is based on ANSI X3.4, CSA Z243.4, ISO DIS 646, CCITT S.100, and ISO DIS 6937.

Where the coded representation consists of two bit combinations (e.g., lower case a with acute accent: 4/2 6/1), the left bit combination shall precede the right one in the information interchange.

Table 20

Latin Alphabetic Characters

Graphic	Name or Description	Coded Representation
a	lower case a	6/1
A	upper case A	4/1
á	lower case a with acute accent	S 4/2 6/1
Á	upper case A with acute accent	S 4/2 4/1
à	lower case a with grave accent	S 4/1 6/1
À	upper case A with grave accent	S 4/1 4/1
â	lower case a with circumflex accent	S 4/3 6/1
Â	upper case A with circumflex accent	S 4/3 4/1
ä	lower case a with diaeresis or umlaut mark	S 4/8 6/1
Ä	upper case A with diaeresis or umlaut mark	S 4/8 4/1
ã	lower case a with tilde	S 4/4 6/1
Ã	upper case A with tilde	S 4/4 4/1
ă	lower case a with breve	S 4/6 6/1
Ă	upper case A with breve	S 4/6 4/1
å	lower case a with ring	S 4/10 6/1
Å	upper case A with ring	S 4/10 4/1
ā	lower case a with macron	S 4/5 6/1
Ā	upper case A with macron	S 4/5 4/1
ą	lower case a with ogonek	S 4/14 6/1
Ą	upper case A with ogonek	S 4/14 4/1

(Continued)

Table 20 (Continued)

Graphic	Name or Description	Coded Representation
æ	lower case æ dipthong	S 7/1
Æ	upper case Æ dipthong	S 6/1
b	lower case b	6/2
B	upper case B	4/2
c	lower case c	6/3
C	upper case C	4/3
ć	lower case c with acute accent	S 4/2 6/3
Ć	upper case C with acute accent	S 4/2 4/3
ĉ	lower case c with circumflex accent	S 4/3 6/3
Ĉ	upper case C with circumflex accent	S 4/3 4/3
č	lower case c with caron	S 4/15 6/3
Č	upper case C with caron	S 4/15 4/3
ċ	lower case c with dot	S 4/7 6/3
Ċ	upper case C with dot	S 4/7 4/3
ç	lower case c with cedilla	S 4/11 6/3
Ç	upper case C with cedilla	S 4/11 4/3
d	lower case d	6/4
D	upper case D	4/4
ď or ḋ	lower case d with caron	S 4/15 6/4
Ď	upper case D with caron	S 4/15 4/4

(Continued)

Table 20 (Continued)

Graphic	Name or Description	Coded Representation
đ	lower case d with stroke	S 7/2
Đ	upper case D with stroke, Icelandic eth	S 6/2
ð	lower case eth, Icelandic	S 7/3
e	lower case e	6/5
E	upper case E	4/5
é	lower case e with acute accent	S 4/2 6/5
É	upper case E with acute accent	S 4/2 4/5
è	lower case e with grave accent	S 4/1 6/5
È	upper case E with grave accent	S 4/1 4/5
ê	lower case e with circumflex accent	S 4/3 6/5
Ê	upper case E with circumflex accent	S 4/3 4/5
ë	lower case e with diaeresis or umlaut mark	S 4/8 6/5
Ë	upper case E with diaeresis or umlaut mark	S 4/8 4/5
ě	lower case e with caron	S 4/15 6/5
Ě	upper case E with caron	S 4/15 4/5
ė	lower case e with dot	S 4/7 6/5
Ė	upper case E with dot	S 4/7 4/5
ē	lower case e with macron	S 4/5 6/5
Ē	upper case E with macron	S 4/5 5/5
ę	lower case e with ogonek	S 4/14 6/5
Ę	upper case E with ogonek	S 4/14 4/5

(Continued)

Table 20 (Continued)

Graphic	Name or Description	Coded Representation
f	lower case f	6/6
F	upper case F	4/6
g	lower case g	6/7
G	upper case G	4/7
ǵ	lower case g with acute accent	S 4/2 6/7
ĝ	lower case g with circumflex accent	S 4/3 4/7
Ĝ	upper case G with circumflex accent	S 4/3 6/7
ğ	lower case g with breve	S 4/6 6/7
Ğ	upper case G with breve	S 4/6 4/7
ġ	lower case g with dot	S 4/7 6/7
Ġ	upper case G with dot	S 4/7 4/7
Ģ	upper case G with cedilla	S 4/11 4/7
h	lower case h	6/8
H	upper case H	4/8
ĥ	lower case h with circumflex accent	S 4/3 6/8
Ĥ	upper case H with circumflex accent	S 4/3 4/8
ħ	lower case h with stroke	S 7/4
Ħ	upper case H with stroke	S 6/4
i	lower case i	6/9
I	upper case I	4/9
í	lower case i with acute accent	S 4/2 6/9
Í	upper case I with acute accent	S 4/2 4/9

(Continued)

Table 20 (Continued)

Graphic	Name or Description	Coded Representation
ì	lower case i with grave accent	S 4/1 6/9
Ì	upper case I with grave accent	S 4/1 4/9
î	lower case i with circumflex accent	S 4/3 6/9
Î	upper case I with circumflex accent	S 4/3 4/9
ï	lower case i with diaeresis or umlaut mark	S 4/8 6/9
Ï	upper case I with diaeresis or umlaut mark	S 4/8 4/9
ĩ	lower case i with tilde	S 4/4 6/9
Ĩ	upper case I with tilde	S 4/4 4/9
İ	upper case I with dot	S 4/7 4/9
ī	lower case i with macron	S 4/5 6/9
Ī	upper case I with macron	S 4/5 4/9
į	lower case i with ogonek	S 4/14 6/9
Į	upper case I with ogonek	S 4/14 4/9
ij	lower case ij ligature	S 7/6
IJ	upper case IJ ligature	S 6/6
ı	lower case i without dot	S 7/5
j	lower case j	6/10
J	upper case J	4/10
ĵ	lower case j with circumflex accent	S 4/3 6/10
Ĵ	upper case J with circumflex accent	S 4/3 4/10
k	lower case k	6/11
K	upper case K	4/11

(Continued)

Table 20 (Continued)

Graphic	Name or Description	Coded Representation
ķ	lower case k with cedilla	S 4/11 6/11
Ķ	upper case K with cedilla	S 4/11 4/11
ĸ	lower case k, Greenlandic	S 7/0
l	lower case l	6/12
L	upper case L	4/12
ĺ	lower case l with acute accent	S 4/2 6/12
Ĺ	upper case L with acute accent	S 4/2 4/12
ľ or l'	lower case l with caron	S 4/15 6/12
Ľ or L	upper case L with caron	S 4/15 4/12
ļ	lower case l with cedilla	S 4/11 6/12
Ļ	upper case L with cedilla	S 4/11 4/12
ł	lower case l with stroke	S 7/8
Ł	upper case L with stroke	S 6/8
ŀ	lower case l with middle dot	S 7/7
Ŀ	upper case L with middle dot	S 6/7
m	lower case m	6/13
M	upper case M	4/13
n	lower case n	6/14
N	upper case N	4/14
ń	lower case n with acute accent	S 4/2 6/14
Ń	upper case N with acute accent	S 4/2 4/14

(Continued)

Table 20 (Continued)

Graphic	Name or Description	Coded Representation
ñ	lower case n with tilde	S 4/4 6/14
Ñ	upper case N with tilde	S 4/4 4/14
ň	lower case n with caron	S 4/15 6/14
Ň	upper case N with caron	S 4/15 4/14
ņ	lower case n with cedilla	S 4/11 6/14
Ņ	upper case N with cedilla	S 4/11 4/14
ŋ	lower case eng, Lapp	S 7/14
Ŋ	upper case eng, Lapp	S 6/14
n	lower case n with apostrophe	S 6/15
o	lower case o	6/15
O	upper case O	4/15
ó	lower case o with acute accent	S 4/2 6/15
Ó	upper case O with acute accent	S 4/2 4/15
ò	lower case o with grave accent	S 4/1 6/15
Ò	upper case O with grave accent	S 4/1 4/15
ô	lower case o with circumflex accent	S 4/3 6/15
Ô	upper case O with circumflex accent	S 4/3 4/15
ö	lower case o with diaeresis or umlaut mark	S 4/8 6/15
Ö	upper case O with diaeresis or umlaut mark	S 4/8 4/15
õ	lower case o with tilde	S 4/4 6/15
Õ	upper case O with tilde	S 4/4 4/15

(Continued)

Table 20 (Continued)

Graphic	Name or Description	Coded Representation
ő	lower case o with double acute accent	S 4/13 6/15
Ő	upper case O with double acute accent	S 4/13 4/15
ō	lower case o with macron	S 4/5 6/15
Ō	upper case O with macron	S 4/5 4/15
œ	lower case œ ligature	S 7/10
Œ	upper case Œ ligature	S 6/10
ø	lower case o with slash	S 7/9
Ø	upper case O with slash	S 6/9
p	lower case p	7/0
P	upper case P	5/0
q	lower case q	7/1
Q	upper case Q	5/1
r	lower case r	7/2
R	upper case R	5/2
ŕ	lower case r with acute accent	S 4/2 7/2
Ŕ	upper case R with acute accent	S 4/2 5/2
ř	lower case r with caron	S 4/15 7/2
Ř	upper case R with caron	S 4/15 5/2
ŗ	lower case r with cedilla	S 4/11 7/2
Ŗ	upper case R with cedilla	S 4/11 5/2
s	lower case s	7/3
S	upper case S	5/3

(Continued)

Table 20 (Continued)

Graphic	Name or Description	Coded Representation
ś	lower case s with acute accent	S 4/2 7/3
Ś	upper case S with acute accent	S 4/2 5/3
ŝ	lower case s with circumflex accent	S 4/3 7/3
Ŝ	upper case S with circumflex accent	S 4/3 5/3
š	lower case s with caron	S 4/15 7/3
Š	upper case S with caron	S 4/15 5/3
ş	lower case s with cedilla	S 4/11 7/3
Ş	upper case S with cedilla	S 4/11 5/3
ß	lower case sharp s, German	S 7/11
t	lower case t	7/4
T	upper case T	5/4
ť or t'	lower case t with caron	S 4/15 7/4
Ť	upper case T with caron	S 4/15 5/4
ţ	lower case t with cedilla	S 4/11 7/4
Ţ	upper case T with cedilla	S 4/11 5/4
ŧ	lower case t with stroke	S 7/13
Ŧ	upper case T with stroke	S 6/13
þ	lower case thorn, Icelandic	S 7/12
Þ	upper case thorn, Icelandic	S 6/12
u	lower case u	7/5
U	upper case U	5/5

(Continued)

Table 20 (Continued)

Graphic	Name or Description	Coded Representation
ú	lower case u with acute accent	S 4/2 7/5
Ú	upper case U with acute accent	S 4/2 5/5
ù	lower case u with grave accent	S 4/1 7/5
Ù	upper case U with grave accent	S 4/1 5/5
û	lower case u with circumflex accent	S 4/3 7/5
Û	upper case U with circumflex accent	S 4/3 5/5
ü	lower case u with diaeresis or umlaut mark	S 4/8 7/5
Ü	upper case U with diaeresis or umlaut mark	S 4/8 5/5
ũ	lower case u with tilde	S 4/4 7/5
Ũ	upper case U with tilde	S 4/4 5/5
ŭ	lower case u with breve	S 4/6 7/5
Ŭ	upper case U with breve	S 4/6 5/5
ű	lower case u with double acute accent	S 4/13 7/5
Ű	upper case U with double acute accent	S 4/13 5/5
ů	lower case u with ring	S 4/10 7/5
Ů	upper case U with ring	S 4/10 5/5
ū	lower case u with macron	S 4/5 7/5
Ū	upper case U with macron	S 4/5 5/5
ų	lower case u with ogonek	S 4/14 7/5
Ų	upper case U with ogonek	S 4/14 5/5
v	lower case v	7/16
V	upper case V	5/16

(Continued)

Table 20 (Concluded)

Graphic	Name or Description	Coded Representation
w	lower case w	7/7
W	upper case W	5/7
ŵ	lower case w with circumflex accent	S 4/3 7/7
Ŵ	upper case W with circumflex accent	S 4/3 5/7
x	lower case x	7/8
X	upper case X	5/8
y	lower case y	7/9
Y	upper case Y	5/9
ý	lower case y with acute accent	S 4/2 7/9
Ý	upper case Y with acute accent	S 4/2 5/9
ŷ	lower case y with circumflex accent	S 4/3 7/9
Ŷ	upper case Y with circumflex accent	S 4/3 5/9
ÿ	lower case y with diaeresis or umlaut mark	S 4/8 7/9
Ÿ	upper case Y with diaeresis or umlaut mark	S 4/8 5/9
z	lower case z	7/10
Z	upper case Z	5/10
ź	lower case z with acute accent	S 4/2 7/10
Ź	upper case Z with acute accent	S 4/2 5/10
ž	lower case z with caron	S 4/15 7/10
Ž	upper case Z with caron	S 4/15 5/10
ż	lower case z with dot	S 4/7 7/10
Ż	upper case Z with dot	S 4/7 5/10

Table 21

Decimal Digits

Graphic	Name or Description	Coded Representation
1	digit 1	3/1
2	digit 2	3/2
3	digit 3	3/3
4	digit 4	3/4
5	digit 5	3/5
6	digit 6	3/6
7	digit 7	3/7
8	digit 8	3/8
9	digit 9	3/9
0	digit 0	3/0

Table 22

Currency Signs

Graphic	Name or Description	Coded Representation
¤	general currency sign	S 2/8
£	pound sign	S 2/3
$	dollar sign	2/4; S 2/4
¢	cent sign	S 2/2
¥	yen sign	S 2/5

Table 23

Punctuation Marks

Graphic	Name or Description	Coded Representation
	space	2/0
!	exclamation point	2/1
¡	inverted exclamation point	S 2/1
"	quotation mark	2/2
'	apostrophe	2/7
(	opening parenthesis (left parenthesis)	2/8
)	closing parenthesis (right parenthesis)	2/9
,	comma	2/12
_	underline	5/15
-	hyphen, minus sign	2/13
.	period (decimal point)	2/14
/	slant (solidus)	2/15
:	colon	3/10
;	semicolon	3/11
?	question mark	3/15
¿	inverted question mark	S 3/15
«	angle quotation marks left	S 2/11
»	angle quotation marks right	S 3/11
‘	single quotation mark left	S 2/9
’	single quotation mark right	S 3/9
“	double quotation marks left	S 2/10
”	double quotation marks right	S 3/10

Table 24

Arithmetic Signs

Graphic	Name or Description	Coded Representation
+	plus sign	2/11
±	plus/minus sign	S 3/1
<	less than sign	3/12
=	equals sign	3/13
>	greater than sign	3/14
÷	divide sign	S 3/8
x	multiply sign	S 3/4

Table 25

Subscripts and Superscripts

Graphic	Name or Description	Coded Representation
1	superscript 1	S 5/1
2	superscript 2	S 3/2
3	superscript 3	S 3/3

Table 26

Fractions

Graphic	Name or Description	Coded Representation
1/2	fraction one-half	S 3/13
1/4	fraction one-quarter	S 3/12
3/4	fraction three-quarters	S 3/14
1/8	fraction one-eighth	S 5/12
3/8	fraction three-eighths	S 5/13
5/8	fraction five-eighths	S 5/14
7/8	fraction seven-eighths	S 5/15

Table 27

Miscellaneous Symbols

Graphic	Name or Description	Coded Representation
#	number sign	2/3; S 2/6
%	per cent sign	2/5
&	ampersand	2/6
*	asterisk	2/10
@	commercial at	4/0
[	opening bracket (left square bracket)	5/11
\	reverse slant (reverse solidus)	5/12
]	closing bracket (right square bracket)	5/13
{	opening brace (left curly bracket)	7/11
|	vertical line	7/12
}	closing brace (right curly bracket)	7/13
µ	micro sign	S 3/5
Ω	ohm sign	S 6/0
°	degree sign	S 3/0
º	masculine ordinal indicator	S 6/11
ª	feminine ordinal indicator	S 6/3
§	section sign	S 2/7
¶	paragraph sign, pilcrow	S 3/6
·	middle dot	S 3/7
←	leftward arrow	S 2/12
→	rightward arrow	S 2/14
↑	upward arrow	S 2/13
↓	downward arrow	S 2/15

(Continued)

Table 27 (Continued)

Miscellaneous Symbols

Graphic	Name or Description	Coded Representation
®	registered sign	S 5/2
©	copyright sign	S 5/3
™	trade mark sign	S 5/4
♪	musical note	S 5/5
`	grave accent	6/0
^	circumflex	5/14
~	tilde	7/14
⁄	diagonal [1]	S 5/8
⧵	reverse diagonal [2]	S 5/9
◢	filled diagonal [3]	S 5/10
◣	filled reverse diagonal [4]	S 5/11
⊞	cross [5]	S 6/5
⊟	full horizontal line [6]	S 5/6
▯	full vertical line [7]	S 5/7
—	horizontal bar [8]	S 5/0

Notes:
(1) The "diagonal" extends from the lower left corner of the character field to the upper right corner of the character field.

(2) The "reverse diagonal" extends from the lower right corner of the character field to the upper left corner of the character field.

(3) The "filled diagonal" is a filled triangle occupying the lower right half of the character field.

(4) The "filled reverse diagonal" is a filled triangle occupying the lower left half of the character field.

(Continued)

Table 27 (Concluded)

Miscellaneous Symbols

(5) The "cross" character is equivalent to overlaying the "full horizontal line" character with the "full vertical line" character.

(6) The "full horizontal line" extends from the middle of the left edge of the character field to the middle of the right edge of the character field.

(7) The "full vertical line" extends from the middle of the bottom edge of the character field to the middle of the top edge of the character field.

(8) The typical graphic representation of "horizontal bar" is a horizontal line at about the middle of a capital letter, ~~and that of "right tine" is a horizontal line at about the top of a capital letter.~~

The rectangles surrounding the characters described by notes (1) to (7) above are for illustrative purposes only and are not part of the graphic symbols.

Table 28

Diacritical Marks

Graphic	Name or Description	Coded Representation
´	acute accent	S 4/2 2/0
`	grave accent	S 4/1 2/0
^	circumflex	S 4/3 2/0
~	tilde	S 4/4 2/0
..	diaeresis or umlaut mark	S 4/8 2/0
¸	cedilla	S 4/11 2/0
˛	ogonek	S 4/14 2/0
˘	breve	S 4/6 2/0
ˇ	caron	S 4/15 2/0
˝	double acute accent	S 4/13 2/0
˙	dot	S 4/7 2/0
¯	macron	S 4/5 2/0
˚	ring	S 4/10 2/0

Note: "Grave accent", "circumflex", and "tilde" are also coded as 6/0, 5/14, and 7/14 respectively.

Table 29

Miscellaneous Non-Spacing Characters

Graphic	Name or Description	Coded Representation
_	non-spacing underline	S 4/12
‾	non-spacing vector overbar	S 4/0
/	non-spacing slant	S 4/9

Note: The graphic representation of "vector overbar" is that of a vector arrow at just above the top of a capital letter. When "non-spacing underline", "vector overbar", and "non-spacing slant" are used, their coded representations precede those of the characters in Tables 20 through 28.

8. Conformance Requirements

8.1 General. The Presentation Level Protocol Syntax described in this Standard provides a ~~coherent, flexible~~ coding scheme by which specific information may be conveyed and correctly interpreted regardless of the hardware dependencies of particular terminal configurations. This provides manufacturers and information providers with the flexibility to develop a ~~broad~~ range of products and service offerings to address different segments of the market, ~~and to cover related areas~~. However, the existence of ranges of service causes problems both for the terminal manufacturer and for the information provider. The manufacturer must ensure that ~~his~~ all products are able to receive and interpret information content that may be beyond the scope of their physical presentation capability, whereas from the information providers' point of view it would be desirable if all terminals displayed the same information identically. These conformance requirements attempt to meet these separate objectives in a reasonable manner.

The conformance requirements specified here can be generally described as defining the rules (syntax) for conforming interchange at the coding interface, as well as the execution (semantics) to be applied by a conforming presentation process. In order to ensure consistent and reliable interchange across the coding interface, it is essential that all interchange be in full compliance with the syntactical rules as defined in this Standard (see Figure 65). It is only in the area of semantic execution that a range of physical properties and display capabilities is possible, and thus requirements are necessary to define conformance.

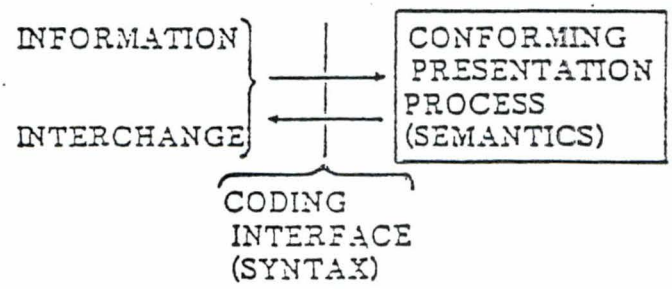


Figure 65

Relationship of Conforming Entities

It is recognized that evolving technology and market forces may determine an implementation appropriate to particular segments of the videotex/teletext industry. These implementations are to be known as Service Reference Models (SRM), which are not part of this Standard. A general SRM for videotex is in Appendix B and should serve as a guideline in defining additional SRM's. Appendix C contains a general SRM for teletext.

8.3.3 A presentation process conforming to this Standard:

8.2 Conforming Interchange

8.2.1 The data stream comprising the exchange of information across the coding interface is subject to certain syntactical restrictions in order to be conforming interchange according to this Standard. Conforming interchange:

(a) Shall be a sequence of 7-bit or 8-bit bit combinations following the syntactic formats described throughout the body of this Standard;

(b) Shall not use any bit combinations allocated by this Standard for any purpose other than that defined in this Standard, unless they represent elements of other code sets that have been invoked by code extension according to ISO 2022, subject to mutual agreement between interchange parties;

For example the cursor position shall be defined before pre

(c) Shall not ~~include sequences that would result~~ in an undefined presentation process or those which are indicated as errors in this Standard; for example, a drawing point shall not be placed outside the unit screen;

(d) Shall immediately follow sequences that result in an implementation dependent state of the presentation process, with those sequences which will re-establish the presentation process to a known state. For example, ~~proportional spacing~~ is implementation dependent. Conforming interchange shall re-establish the position of the cursor and the graphic drawing point if tied to the cursor.

the positioning with respect to

automatic APR APD, word wrap and proportional sp

8.3 Conforming Presentation Process

A

8.3.2 A presentation process conforming to this Standard shall be capable of receiving all conforming interchange (according to Clause 8.2). ~~and~~

and recognizing

(a) Shall execute all code extension sequences defined in this Standard;

(b) Shall execute and present all 94 characters of the Primary set consistent with the text attributes implemented;

(c) Shall execute ~~present~~ all remaining characters of the graphic character repertoire either exactly or in an approximate form or as a symbol that cannot be confused with any character in the graphic character repertoire;

and present them

(d) Shall execute and present all geometric graphic primitives (i.e., POINT, LINE, ARC, RECTANGLE, POLYGON, and INCREMENTAL) and shall execute the control functions RESET, DOMAIN, and WAIT;

(f) ~~Not~~ Shall execute all other control codes from the PDI set, and, consistently with the attributes implemented, shall present them;

(g) ~~Not~~ Shall execute and present all 65 codes of the mosaic set;

(e) Shall execute all discrete values of the attribute control functions which affect the recognition and execution of subsequent code sequences, i.e., single-value operand length (1-4), multi-value operand length (1-8), dimensionality (2,3) and color modes (0, 1 and 3).

except ... one-way services where the following may not be required: FLASH CURSOR, STEADY CURSOR, CURSOR OFF, PROTECT and UNPROTECT. The presentation of the C1 control codes is implementation dependent and present except for REPEAT, SELECT TO EOL, SCROLL UP and SCROLL ... implem...

(h) ~~Dgt~~ Shall execute all 96 macros, which are defined according to the procedures of this Standard;

(i) ~~Dm~~ Shall execute and present all 96 DRCS characters, which are dynamically redefinable according to the procedure of this Standard;

(j) ~~Dx~~ Shall execute and present all C0 control functions defined in this Standard;

(k) ~~Dx~~ Shall execute all control codes from the C1 code set as defined in this Standard. ~~Note that the execution of these control codes is dependent upon the attribute implementation as defined in the~~ appropriate ~~SPM.~~

8.3.~~2~~ 4 In a conforming presentation process, attributes may be implemented to a variety of capabilities. The following shall apply:

(a) All parameters for drawing commands or attributes that are specified as continuous variables may be approximated with implementation dependent value or values, one value of which shall be the default value;

(b) All parameters for drawing commands or attributes that may take on multiple (two or more) discrete values may be approximated with implementation dependent value or values, one value of which shall be the default value;

(c) For the interpretation of all conforming interchange, the cursor and drawing point positions shall be calculated so as to minimize the accumulation of round-off error.

8.3.~~3~~ 5 The default conditions of the attributes for the coding scheme described in this Standard are summarized in Table 30.

8.3.1 In this clause, the following definitions apply:

(a) Recognize means to determine the syntactic form of a code sequence;

(b) Execute means to process a code sequence to allow the displa... of information conveyed by the code sequence if presented, and by subsequent code sequences as specified in this Standard (e.g., computation of positional information);

(c) Present means to display the information conveyed by a code sequence, and in the case of a control function to display subse... information affected by the control function.

Table 30

Default Conditions

Parameters	Values
Color mode:	0
In-use color:	White
Single-value length (color map address):	1 byte
Multi-value length (coordinate data):	3 bytes
Dimensionality:	2 dimensions
Logical pel size:	dx = +0, dy = +0
Text rotation:	0° (relative to horizontal)
Character path:	To the right
Inter-character spacing:	1 (width of current character field)
Inter-row spacing	1 (height of current character field)
Move parameters:	Cursor and graphics drawing point move together.
Cursor style:	Underscore
Cursor display:	Off (invisible)
Character field dimensions:	dx = approximately 0.025 (i.e. dx = approximately 0.00000110 binary) dy = approximately 0.04 (i.e. dy = approximately 0.00001010 binary)
Line texture:	Solid
Texture pattern:	Solid
Highlight:	No highlight
Texture mask size:	dx = approximately 0.025 (i.e. dx = approximately 0.00000110 binary) dy = approximately 0.04 (i.e. dy = approximately 0.00001010 binary)
Underline mode:	Off
Word wrap mode:	Off
Scroll mode:	Off
Blink condition:	Off
Active field:	The unit screen, protected
cursor position and graphics drawing position	undefined

9. Enhanced Capabilities

9.1 The future addition of enhanced capabilities is accommodated by this syntax via the code extension techniques defined in Clause 4. New features could be grouped into logically consistent classes, which would then be used to define entire additional code sets that would be added to the existing repertory. The addition of new functionality within the G- and C-sets described herein beyond that already defined ~~is considered to be in error~~

For example to PDI commands which are specified with fixed sequences (where additional operand data bytes are specifi
to be ignored), is reserved for future compatible extension to this standard.

Appendix A

Coordinate System Concepts

Note: This Appendix is not a mandatory part of this Standard.

The Presentation Level Protocol Syntax (PLPS) makes use of the concept of an abstract Cartesian coordinate system upon which all pictorial, textual, and incremental drawing operations are defined. Normalized coordinates are used to define a unit screen. In two dimensions, the unit screen consists of a square area whose points along the X and Y axes range from 0 (inclusive) to 1 (non-inclusive). In three dimensions, the unit screen consists of a cube whose points along the X, Y, and Z axes also range from 0 to 1, where 0 is defined as the farthest point along the Z axis from the viewer. No other details of the three dimensional mode have been defined.

Drawing commands and other operations make use of coordinate specifications that are binary fractions of the unit screen. Drawing specifications outside the unit screen either directly specified in the negative coordinate space below or to the left of zero or specified by relative displacements outside the unit screen are in error.

an implementation dependent rounding

Coordinate specifications may be described to several levels of accuracy due to their representation as fractions. Unnecessary least significant bits are eliminated by ~~a truncation~~ process when the specification is to a greater accuracy than can be handled by the terminal. Drawing commands are normally defined using 8 bits plus sign occupying 3 bytes of data to define an X, Y, coordinate position. This implies a minimum specifiable binary fraction of 1/256 of the unit screen. By defining additional data bytes under control of the DOMAIN command, greater accuracy may be specified. This numbering system in no way restricts the terminal manufacturer from implementing any resolution he feels is cost effective at the basic hardware level. The number system merely describes the manner by which pictures are described so that there is display hardware independence in the PLPS code. The manufacturer would convert coordinate information from the PLPS coordinate system to his working coordinate system. If his hardware number system bears a simple relation to the PLPS number system, this would be a trivial conversion.

This Standard is written in the abstract terms of the unit drawing area only, in order not to eliminate any possible implementation; however, this abstraction can cause confusion for the implementor and for the information provider. Since the Service Reference Model for a particular service defines certain parameters such as resolution, it becomes possible to greatly clarify the actual display results that would occur on various types of hardware. An example is developed below that shows the physical display parameters that would result from the choice of a bit plane memory implementation with 256 pixels in the X axis and 4 bit planes of memory. References are also given as to how this would be implemented on a character generator memory terminal.

The Standard indicates that the exact registration of the pictorial, textual, and photographic drawing techniques is not guaranteed. This is because for certain display resolutions or when using certain display technologies, particularly the most cost effective ones, the number of physical display pixels does not exactly divide into the number of logical pels specified, that is, quantization error may result. Quantization error causes an effect similar to moiré patterns in optics, which can cause a severe degradation of a picture. For example, it is impossible to map a 10 x 10 array of points to 13 x 13 array of physical pixels. The doubling up of points would totally distort the pattern.

For example, examining the case of 256 physical pixels of resolution in each axis of a bit plane memory, one can map the unit screen of 0 (inclusive) to 1 (non-inclusive) to it exactly. The binary representation of fractions map exactly to the pixel locations, with the result that a 45° line is smooth. Pixel elements must be square or very nearly square for there to be no appearance of geometric distortion. Circles must appear round. It is not possible to scale the coordinate data in either axis by other than simple ratios or quantization error will result. This means that in the vertical direction the physical pixels should align with the scan lines of the television display.

The text format has as its default value 40 characters per row and 20 rows guaranteed within the display area of the rectangular 4:3 aspect ratio television display screen. The aspect ratio applies to the outer bounds of the television signal video area, and safety margining of approximately 10% on the top, bottom, and both sides must be left by the terminal in order to accommodate the overscan of the television sets. This is the Society of Motion Picture and Television Engineers (SMPTE) (RP 27.3) "safe title area". The number 40 does not divide exactly into 256 and so the 40 character positions must be defined to fit on display screen in slightly less than 256 pixel elements in order to avoid quantization error effects. This means that width of the character field would be 6/256.

Psychological studies have shown that the most readable characters in this size range are 6 pixels by 10 pixels. This, then, means that in the vertical direction 200 scan lines are required to produce 20 rows. A 21st service row may occupy 10 additional scan lines.

If pictures are drawn as pixel patterns using the INCREMENTAL POINT command, they cannot be scaled except by simple integer exact divisions or multiplications without resulting in quantization error. This means that such pictures may be reduced in size in a transformation from one piece of apparatus of one physical display density to another with a different and not simply related resolution. A similar effect causes the scaling of DRCS characters and definable texture masks to be limited to integer exact divisions or multiplications. Such scaling is required when a DRCS or texture mask is described by a character field that is greater than the physical limit implied by the memory size available for redefinable functions.

The physical parameters for the display apparatus used in this example are:

(a) 256 pixels in the X axis by 200 pixels in the Y axis;

(b) An aspect ratio of 0.78125;

(c) Default character field physical dimensions of 6/256 in X by 10/256 in Y;

(d) Character column zero is justified to the left edge of the unit screen.

Character sizes other than the default "NORMAL" text can also fit within this display area. Text of 32 character positions and 16 rows divides exactly into the 256 pixel positions in X and makes use of 192 scan lines in Y, with resulting physical character field dimensions of 12/256 in Y by 8/256 in X. Text of 40 character positions and 24 rows would result in 6/256 in X by 8/256 in Y.

The use of character generator hardware would permit any one particular character density to be displayed; however, the same quantization effects would affect the arbitrary display of pictorial information. The numbers defined in the above example would also apply.

As another example, 240 physical pixels of resolution may be used over the full 0 to 1 range in the X axis, along with 200 pixels in the Y axis. The aspect ratio of 0.78125 remains the same as in the previous example. The default character field physical dimensions are 6/240 in the X axis by 10/256 in the Y axis. This yields exactly 40 columns of characters. The transformation from binary fraction specifications of coordinates into 240 physical pixels may not be exact.

Since the numbers used in the above examples may apply to the SRM(s), they can be used as a guideline for information providers to define pictures.

Appendix B

A General Service Reference Model (SRM)
for Videotex

Note: This Appendix is not a mandatory part of this Standard.

The Service Reference Model (SRM) outlines a set of specific implementation parameters for a particular service. It defines the functionality available to the information provider and service operator for the generation and display of information. ~~This is a specification of the features that may be used by the information provider for the given service.~~ It is the maximum set of features that may be used to create information. If additional features are used, they may not be interpreted exactly as the information provider intended them on all terminal or decoder apparatus. However, in closed applications, by mutual agreement of the sender and receiver, additional functionality may be used and these features would be approximated in the manner defined by this Standard on terminals implementing to the Service Reference Model.

A manufacturer may implement features to a greater or lesser extent than defined in the Service Reference Model and still conform to the PLPS, just as a monochrome television set conforms to the National Television Standards Committee (NTSC) color television standard. If a user buys a terminal that conforms to the PLPS yet does not implement the complete functionality of the Service Reference Model, some information may be lost just as a user who buys a monochrome television receiver loses the color information. Analogously, in both cases these standards have been organized in such a way that the essence of the information is preserved.

The parameters listed in Table B1 have been identified as necessary to define a Service Reference Model for videotex (interactive). The conformance section (Clause 8) of this Standard indicates which functions must be presented exactly and which can be presented in an approximate form.

Table B1

General Videotex SRM Parameters

Presentation Level Protocol Syntax and Semantics		Service Reference Model Requirements Videotex
1.	Code extension sequence (Ref. 8.3.1(a))	- execute all code extension sequences defined in this Standard.
2.	Primary set (Ref. 8.3.1(b))	- all 94 primary characters must be implemented. - approximation is not permitted. - character font depends on implementation.
3.	All remaining characters of the graphic character repertoire (Ref. 8.3.1(c))	- all remaining characters of the graphic repertoire (see Clause 7) must be implemented. - approximation is not permitted. - character font depends on implementation.
4.	Geometric graphic primitives and control functions (PDI set) (Ref. 8.3.1(d))	- as specified in Clause 8.3.1(d). - the minimum duration of a wait interval for a WAIT operand of zero (see Clause 5.3.2.8.2) is 1/30 second.
5.	PDI attributes (Ref. 8.3.1(e))	
5.1	~~Logical pel~~ DOMAIN	logical pel - the full range of heights and widths consistent with the physical resolution (see Item 11).
5.2	TEXT	- the positions of the cursor and dy point must be computed and maintai at least nine binary places (1/512 of t
5.2.1	TEXT sizes	- the following text sizes: NORMAL, MEDIUM, DOUBLE HEIGHT, DOUBLE SIZE (see Appendix A).
5.2.2	Character field sizes	- all character field sizes greater than or equal to dx = 6/256 and dy = 8/256 (see Appendix A) consistent with the physical resolution, which includes the following character display formats (column x row): 40 x 24, 40 x 20, 40 x 10, 32 x 16, 20 x 10.
5.2.3	Character rotation	- all 4 directions (i.e., 0°, 90°, 180° and 270°).

Table B1 (Continued)

Presentation Level Protocol Syntax and Semantics	Service Reference Model Requirements Videotex
5.2.4 Character path movement	- all 4 directions (i.e. right, left, up and down).
5.2.5 Inter-character spacing	- 1, 1.25, 1.5, and proportional spacing consistent with the physical resolution. - spacing of proportional spacing depends on implementation, as font is not defined.
5.2.6 Inter-row spacing	- 1, 1.25, 1.5 and 2.
5.2.7 Move attributes	- all four attributes (i.e., move together, cursor leads, drawing point leads, and move independently).
5.2.8 Cursor style	- all four styles (i.e., underscore, block, cross-hair, and custom).
5.3 TEXTURE	
5.3.1 Line texture	- all four textures (i.e., solid, dotted, dashed, and dotted-dashed).
5.3.2 Texture pattern	- all four defined masks (i.e., solid, vertical hatching, horizontal hatching, and cross-hatching) and four programmable masks (see Appendix A).
5.3.3 Highlight	- in accordance with Clause 5.3.2.4.3 and in all color modes.
5.4 COLOR	- all color modes (i.e., 0, 1, and 2). - minimum of 16 simultaneous colors out of a minimum set of 512 different possible colors. Note that the exact values of the default colors are not guaranteed due to the round-off errors in the color equations (see Clause 5.3.2.5.1). - color attribute of the border is not guaranteed.
5.5 BLINK	- BLINK commands from PDI set and C1 set. - minimum number of blink process = $2^N = 16$ where N = number of bits of the color map address.

MK3 CONFIGURATION:

1 [0] B0 0] PARITY
0 B1 0]
0 B2 0 0=NO LOCAL ECHO
0 B3 0 DIP/$\overline{DB25}$
0 B4 0 ALPHA/$\overline{GRAPH}$
0 B5 0
0 B6 0 MAP
0 B7 0 MAP

B1	B0	
0	0	ODD
0	1	EVEN
1	0	MARK
1	1	SPACE

DIP 8E

PROPER SETUP: SHORT 2-15 EVEN PARITY
3-14 NO LOCAL ECHO
4-13 DB25 = I/P
OTHERS = OPEN

BAUD RATE (DIP 6):

75	1 [0]	0	150
300	0	0	600
1200	0	0	RXCLK
2400	0	0	4800
9600	0	0	N/C
N/C	0	0	N/C
RX CLK	0	0	TX CLK
N/C	0	0	N/C

MKJ CONFIGURATION:

1 [O] B0 O] PARITY
O B1 O]
O B2 O O=NO LOCAL ECHO
O B3 O DIP/$\overline{DB25}$
O B4 O ALPHA/$\overline{GRAPH}$
O B5 O
O B6 O MAP
O B7 O MAP

B1	B0	
0	0	ODD
0	1	EVEN
1	0	MARK
1	1	SPACE

DIP 5E

PROPER SETUP: SHORT 2-15 EVEN PARITY
3-14 NO LOCAL ECHO
4-13 DB25 = I/P
OTHERS = OPEN

BAUD RATE (DIP 6):

1
75 [O] O 150
300 O O 600
1200 O O RXCLK
2400 O O 4800
9600 O O N/C
N/C O O N/C
RX CLK O O TX CLK
N/C O O N/C

MK3 CONFIGURATION:

1 [O] B0 O] PARITY
O B1 O]
O B2 O 0=NO LOCAL ECHO
O B3 O DIP/DB25
O B4 O ALPHA/GRAPH
O B5 O
O B6 O MAP
O B7 O MAP

B1	B0	
0	0	ODD
0	1	EVEN
1	0	MARK
1	1	SPACE

DIP 3E

PROPER SETUP: SHORT 2-15 EVEN PARITY
3-14 NO LOCAL ECHO
4-13 DB25 = DIP
OTHERS OPEN

BAUD RATE (DIP 6):

75	1 [O]	O 150
300	O	O 600
1200	O	O RXCLK
2400	O	O 4800
9600	O	O N/C
N/C	O	O N/C
RXCLK	O	O TX CLK
N/C	O	O N/C

MK3 CONFIGURATION:

1	☒	B0	C	] PARITY	B1 B0
	○	B1	○		0 0 ODD
	○	B2	○	0 = NO LOCAL ECHO	0 1 EVEN
	○	B3	○	DIP/$\overline{DB25}$	1 0 MARK
	○	B4	○	ALPHA/$\overline{GRAPH}$	1 1 SPACE
	○	B5	○		
	○	B6	○	MAP	
	○	B7	○	MAP	

DIP 5E

PROPER SETUP: SHORT 2-15 EVEN PARITY
3-14 NO LOCAL ECHO
4-13 DB25 = I/P
OTHERS = OPEN

BAUD RATE (DIP 6):

75	1 ☒	○	150
300	○	○	600
1200	○	○	RXCLK
2400	○	○	4800
9600	○	○	N/C
N/C	○	○	N/C
RXCLK	○	→	TX CLK
N/C	○	○	N/C

MK3 CONFIGURATION:

1 ■ B0 ○ ┐ PARITY
○ B1 ○ ┘
○ B2 ○ 0=NO LOCAL ECHO
○ B3 ○ DIP/$\overline{DB25}$
○ B4 ○ ALPHA/$\overline{GRAPH}$
○ B5 ○
○ B6 ○ MAP
○ B7 ○ MAP

B1	B0	
0	0	ODD
0	1	EVEN
1	0	MARK
1	1	SPACE

DIP 3E

PROPER SETUP: SHORT 2-15 EVEN PARITY
3-14 NO LOCAL ECHO
4-13 DB25 = I/P.
OTHERS = OPEN

BAUD RATE (DIP 6):

	1	
75	■	○ 150
300	○	○ 600
1200	○	○ RXCLK
2400	○	○ 4800
9600	○	○ N/C
N/C	○	○ N/C
RX CLK	○	○ TX CLK
N/C	○	○ N/C

MK3 CONFIGURATION:

1 ☒ B0 ○ ┐
○ B1 ○ ├ PARITY
○ B2 ○ 0 = NO LOCAL ECHO
○ B3 ○ DIP/$\overline{DB25}$
○ B4 ○ ALPHA/$\overline{GRAPH}$
○ B5 ○
○ B6 ○ MAP
○ B7 ○ MAP

B1	B0	
0	0	ODD
0	1	EVEN
1	0	MARK
1	1	SPACE

DIP 5E

PROPER SETUP: SHORT 2-15 EVEN PARITY
3-14 NO LOCAL ECHO
4-13 DB25 = I/P
OTHERS = OPEN

BAUD RATE (DIP 6):

75	1 ☒	150
300	○	○ 600
1200	○	○ RXCLK
2400	○	○ 4800
1800	○	○ N/C
N/C	○	○ N/C
RX CLK	○	○ TX CLK
N/C	○	○ N/C

Table B1 (Continued)

Presentation Level Protocol Syntax and Semantics		Service Reference Model Requirements Videotex
6.	Mosaics set (Ref. 8.3.1(f))	- both separated and contiguous mosaics. - approximation is not permitted.
7.	Macro set (Ref. 8.3.1(g))	- minimum 3 Kbytes of shared memory (shared with DRCS, unprotected fields, and other memory intensive processes).
8.	DRCS (Ref. 8.3.1(h))	- minimum 3 Kbytes of shared memory (shared with macros, unprotected fields, and other memory intensive processes).
8.1	DRCS size	- the first DRCS character definition may establish the storage resolution for all subsequently defined DRCS characters until the set is cleared. - all TEXT attributes (i.e., Item 5.2) still apply when defined and displayed.
9.	C0 control set (Ref. 8.3.1(i))	- as specified in Clause 8.3.1(i).
10.	C1 Control set (Ref. 8.3.1(j)	- executes all control codes from C1 code set as defined in this Standard.
10.1	Unprotected field	- only characters may be allowed in unprotected fields for the purpose of editing and subsequent transmission. - a minimum of 40 unprotected fields (based upon the default horizontal character field size) may be available, subject to the memory limitations. - minimum of 3Kbytes of shared memory (shared with DRCS, macros, and other memory intensive processes). - characters may be drawn with a uniform size in any one unprotected field, which is established at the time of the definition of the unprotected field.

Table B1 (Concluded)

Presentation Level Protocol Syntax and Semantics	Service Reference Model Requirements Videotex
	- color is permitted to change between each character field within an unprotected field - no other attributes are assured.
11. Physical display parameters	- minimum resolution in the order of 256 x 200 pixels for the physical display area, resulting in an exact aspect ratio of 0.78125 and maintaining left edge justification of character column zero (see Appendix A).
10.2 SCROLL ON	- no buffering of data scrolled out of the physical display area or active field is required.

Appendix C

A General Service Reference Model (SRM)
for Teletext Services

Note: This Appendix is not a mandatory part of this Standard.

The Service Reference Model for Teletext services outlines the set of specific implementation parameters for the teletext service. A teletext presentation process must meet the conformance requirements specified for conforming interchange in Clause 8.2 as well as the execution of presentation semantics as specified in Clause 8.3. The format of the interactive videotex SRM (Appendix B) will serve as a guideline for the format of the teletext SRM, which is still under study.

The EIA BTS Teletext Steering Committee Special Working Group on NABTS has indicated its intention to submit a teletext SRM to ANSI/CSA. The CVCC Teletext Subcommittee has indicated its intention to work jointly with the above group to this end. The EIA and CVCC have agreed to jointly develop a common teletext SRM for North America.

Appendix D

Service Reference Model (SRM)
for Other Applications

Note: This Appendix is not a mandatory part of this Standard.

Other Service Reference Models (SRMs) may be developed by the appropriate industrial committees and organizations.

Bibliography

"Telidon Videotex Presentation Level Protocol: Augmented Picture Description Instructions," CRC Technical Note No. 709, Department of Communications Canada, C.D. O'Brien, H.G. Bown, J.C. Smirle, Y.F. Lum and J.Z. Kukulka, February 1982.

"North American Broadcast Teletext Specification", CBS Television Network, June 1981.

"European Interactive Videotex Service Display Aspects and Transmission Coding," CEPT Sub-Working Group CD/SE Recommendation No. T/CD 6-1, June 1981.

"Bell System Videotex Standard Presentation Level Protocol", American Telephone & Telegraph Company, May 1981.

"Picture Description Instructions (PDI) for the Telidon Videotex Systems", CRC Technical Note No. 699, Department of Communications, Canada, H.G. Bown, C.D. O'Brien, W. Sawchuk, and J.E. Storey, November 1979.

"Status Report of the Graphics Standards Planning Committee", Computer Graphics quarterly report of SIGGRAPH-ACM, Vol. 13, No. 3, August 1979.

"On the Generation and Representation of Line Drawings", CRC Technical Note No. 589, Department of Communications, Canada, H.G. Bown and P.E. Allard, February 1978.

"SMPTE Recommended Practice, Specifications for Safe Action and Safe Title Areas, Test Pattern for Television Systems", Society of Motion Picture and Television Engineers, RP 27.3, 1972