

In the geometric display mode, the following values are used:

6/8 geometric profile x 1

6/9 geometric profile x 2

In the ASCII display mode, the following values are used:

7/14 x/y

as specified in Part 10.

The following restricting rules apply to the use of the switching sequence:

- The switching command is recognized only inside an alphamosaic VPDE.
- The switching command must contain one and only one alphamosaic or ASCII profile. This is the default profile after the switching command.

The profile selection is terminated either:

- by selection of another profile environment; or
- by following the rules of ISO 2022 (see clause 5).

NOTE – When the Profile switching mechanisms is executed no assumption can be made about the subsequent contents of the screen.

## **Annex D**

### **International Interworking for Videotex Services Data Syntax III**

(This annex forms an integral part of this Recommendation)

#### **Preface**

This Data Syntax specifies the coding scheme to be used in videotex and teletext services. Videotex and teletext services are two-way and one-way services, respectively, providing users with access to “pages” or “frames” that include alphanumeric and pictorial information.

This Data Syntax provides for the “blind interchange” that is desirable for electronic media envisioned for videotex and teletext. Blind interchange means that the sender and receiver of the information do not need any prior agreements or negotiation dialogue in order to meaningfully interchange information; they need only to agree to conform to the Data Syntax.

The conformance requirements specified 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. These conformance requirements are carefully specified so that a number of different implementation technologies can be used. For example, the body of the Data Syntax does not specify the resolution of the display. It is intended that the same interchange can be presented on current TV sets as well as higher resolution devices.

In order to further define sets of specific implementation parameters for videotex and teletext, two Service Reference Models (SRMS) for videotex and teletext are included in Appendix IV. An SRM specifies the minimum implementation of a receiving device and the maximum implementation to be assumed by an information provider. An implementation which meets the requirements of an SRM also conforms to the Data Syntax.

In this Data Syntax, use of the words “shall”, “should”, and “may” represent requirements, recommendations, and options, respectively.

## D.1 Scope

**D.1.1** This Data Syntax describes the formats, rules, and procedures for the encoding of alphanumeric text and pictorial information for videotex and teletext applications. This Data Syntax is based on the architecture defined in ISO's and CCITT's multilayered reference model of open systems interconnection. This Data Syntax defines a specific data syntax for use by OSI presentation layer protocols and some specific semantics for use at the application layer in videotex and teletext applications.

**D.1.2** The basic coding scheme is built upon the framework established by Recommendation S.100 (1980) (International Information Exchange for Interactive Videotex) of the International Telegraph and Telephone Consultative Committee (CCITT). Operation in both a 7-bit and an 8-bit environment is accommodated. Alphanumeric text, a set of supplementary characters, and a 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 (1980). The geometric primitives are compatible enhancements to the picture description instructions (POI's) defined in the alphegeometric option of CCITT Recommendation S.100 (1980). Additional capabilities include colour 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.

## D.2 Definitions

**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 non-negative absolute coordinate specifications lie within the unit screen.

**alphanumeric text:** Means the written form of languages, comprising 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 Recommendation, 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:** Means 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 display area.

**C-set; control set:** Means the two control sets, C0 and C1, each comprising 32 character positions arranged in two columns of 16.

**character field:** Means the rectangular area within which a character is displayed.

**code extension:** Means the 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:** Means an interface through which coded bit combinations are passed between receiving equipment and communication media.

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

**colour map:** Means a look-up table that is used during scan conversion of the digital image that converts colour map addresses into actual colour values.

**colour value:** Means an entry in a colour map that indicates the actual colour of the pixel to be displayed.

**composite symbol:** Means 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.

**consistent with the physical resolution:** Means the position of the display information is calculated to sufficient precision that it is displayed within one pixel of the true position (see Appendix II).

**cursor:** Means a logical indicator (having character field dimensions) 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.

**designate:** Means to identify a given set from the repertory of G-sets as a G0, G1, G2, or G3-sets.

**display area:** Means the rectangular part of the physical display screen in which information coded in conformance to this Recommendation is visibly displayed. The display area does not include the border area.

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

**dynamically redefinable character set (DRCS):** Means a G-set containing definable characters whose patterns can be downloaded from the host.

**escape sequence:** Means a string of two or more bit combinations beginning with the ESCAPE (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 four active G-sets. A two-character escape sequence contains only a final character (F), and is 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 2022-1982.

**final character:** Means the last character of an escape sequence.

**G-set:** Means one of the four sets, G0, G1, G2, and G3, each of which comprises 94 or 96 character positions arranged in six columns of 16.

**G-set repertory:** Means the collection of available code sets that are subject to designation as one of the G-sets.

**geometric graphic primitive:** Means a locally stored picture drawing algorithm that can be called via a specified opcode and associated operand(s).

**graphic character repertoire:** Means the list of graphic characters defined in this Data Syntax, including accented letters and characters obtained by the composition of two or more graphic symbols.

**implementation-dependent:** Means a feature that may be specified more completely in a service reference model or by an implementor, within the constraints imposed by this Data Syntax.

**in-use:** Means the code sets or attributes that will be used to interpret or be applied to subsequently received commands.

**intermediate character:** Means 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:** Means each individual module of the reference model for open systems interconnection (OSI).

**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:** Means a geometric construct associated with the drawing point whose size determines the stroke width of graphics primitives. 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:** Means 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:** Means a rectangular matrix of predefined elements that can be used to construct block-style graphic images.

**nominal black:** Means the colour black (all zeros) in colour mode zero, or the colour that is at colour map address zero in colour modes 1 and 2.

**nominal white:** Means the colour white (all ones) in colour mode zero, or the colour that is at colour map address 011...1 in colour modes 1 and 2.

**non-spacing:** Means a character that does not cause the cursor to be automatically advanced when it is received after the character is displayed.

**opcode:** Means a one-byte character that initials the execution of a locally stored geometric primitive or control operation. An opcode may be followed by zero or more operands.

**operand:** Means 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 picture element (pixel):** Means the smallest displayable unit on a given display device.

**pictorial information:** Means the display information resulting from the application of geometric primitives, mosaics, and DRCS.

**picture description instruction (PDI):** Means a command composed of an opcode followed by zero or more operands that constitutes an executable picture drawing or control command.

**presentation layer:** Means the sixth of seven layers defined by the reference model for open systems interconnection. The use of the presentation layer in this Data Syntax is primarily for the encoding of text, graphic, and display control information.

**protocol:** Means a set of formats, rules, and procedures governing the exchange of information between peer processes at the same layer.

**receiving device:** Means equipment that can receive coded bit combinations by means of, for example, telecommunication or physical interchange of storage media.

**relative coordinates:** Means 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.

**service reference model (SRM):** Means a specification of the minimum set of features that must be implemented by a receiving device in order to meet the requirements for a particular service and the maximum set of features that should be assumed by an information provider when encoding text and pictorial information.

**single shift:** Means 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).

**spacing:** Means a character that causes the cursor to be automatically advanced when it is received after the character is displayed.

**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 (inclusive) to 1 (non-inclusive) in the horizontal (X), vertical (Y), and depth (Z) dimensions. (The last is only declined in three-dimensional mode.)

### **D.3 Reference Publications**

**D.3.1** This Data Syntax refers to the following publications and where reference is made it shall be to the edition listed below, including all revisions published thereto.

#### **D.3.2 ANSI<sup>15)</sup> Standards**

ANSI X3.4-1977

American National Standard Code for Information Interchange (ASCII).

ANSI 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.

ANSI X3.110-1983/CSA T500-1983

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

#### **D.3.3 CSA<sup>16)</sup> 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.

CSA T500-1983/ANSI X3.110-1983

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

#### **D.3.4 CCIR<sup>17)</sup> Report**

957-October, 1981

Characteristics of Teletext Systems, Document 11/5001-E.

#### **D.3.5 CCITT<sup>18)</sup> Recommendations**

F.300-1980

Videotex Service.

S.100-1980

International Information Exchange for Interactive Videotex.

V.3-1972

International Alphabet No. 5

X.200 (Draft)

Reference Model of Open Systems Interconnection for CCITT Applications

#### **D.3.6 Department of Communication, Canada**

Telecommunications Regulatory Service

Broadcast Specification BS-14 June, 1981

#### **D.3.7 EIA<sup>19)</sup>/CVCC<sup>20)</sup> Recommendation 1983,**

North American Basic Teletext Specification (NABTS).

---

15) American National Standards Institute.

16) Canadian Standards Association.

17) International Radio Consultative Committee.

18) International Telegraph and Telephone Consultative Committee.

19) Electronic Industries Association.

20) Canadian Videotex Consultative Committee.

### **D.3.8 ISO<sup>21)</sup> Standards**

646-1983

Information Processing – ISO 7-bit Coded Character Set for Information Interchange.

2022-1982

Information Processing – ISO 7-bit and 8-bit Coded Character Sets – Code Extension Techniques.

2375-1980

Data Processing – Procedure for Registration of Escape Sequences.

DIS<sup>22)</sup> 6937/1-1982

Information Processing – Coded Character Sets for Text Communication – Part 1: General Introduction.

DIS<sup>22)</sup> 6937/2-1982

Information Processing – Coded Character Sets for Text Communication – Part 2: Latin Alphabetic and non-Alphabetic Graphic Characters.

DIS<sup>22)</sup> 7498-1983

Information Processing Systems – Open Systems Interconnection – Basic Reference Manual

## **D.4 Coding Architecture**

### **D.4.1 Reference Model (OSI)**

The coding scheme described in this Data Syntax addresses itself primarily to the presentation layer of the seven layer reference model for open systems interconnections. This reference model is described in CCITT Recommendation X.200/ISO DIS 7498-1983, Information Processing Systems – Open Systems Interconnection – Basic Reference Model (see Appendix I).

### **D.4.2 Presentation Level Overview**

#### **D.4.2.1 General**

This Data Syntax is based on the code extension principles of ISO 2022-1982<sup>23)</sup> and on currently existing national and international Standards and Recommendations. The relationship of the various coding standards and the manner in which they are woven into a unified data syntax is described below.

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 Data Syntax, or for use at other protocol layers, for example, parity.

The character code table is normally represented as a table of 8 columns and 16 rows with bits  $b_7$ ,  $b_6$ , and  $b_5$  addressing the columns and bits  $b_4$ ,  $b_3$ ,  $b_2$ , and  $b_1$  addressing the rows (see Figure D.3). This general format is used throughout this Data Syntax. In diagrams, the bits are numbered  $b_1$  to  $b_8$  with  $b_1$  occupying the least significant position. See Figure D.1. The code table may be sub-divided into different segments as detailed in D.4.3.

The coding of alphanumeric characters is based on CCITT Recommendations V.3 (1972), and S.100 (1980), and ISO 646 (1982) (see D.5.2).

The coding of the pictorial information is based on:

- 1) Picture description instructions (PDI's) (see D.5.3), which are based on an enhancement of the alphageometric option described in CCITT Recommendations S.100 (1980) and F.300 (1980).
- 2) Mosaic set (see D.5.4), which is based on the union of the two mosaic tables described in CCITT Recommendations S.100 (1980) and F.300 (1980).

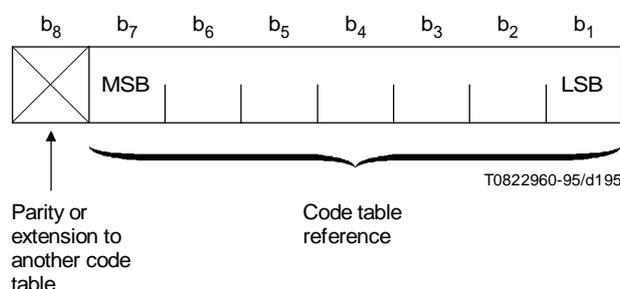
---

21) International Organization for Standardization.

22) DIS means Draft International Standard, which is subject to revision.

23) ISO is developing a draft addendum or revision to ISO 2022-1982 which deals, among other things, with the definition and code extension of 96 character G-sets.

- 3) Macro set (see D.5.5).
- 4) Dynamically redefinable character set (DRCS) (see D.5.6).



**Figure D.1/T.101 – Coding format**

#### D.4.2.2 Coordinate System

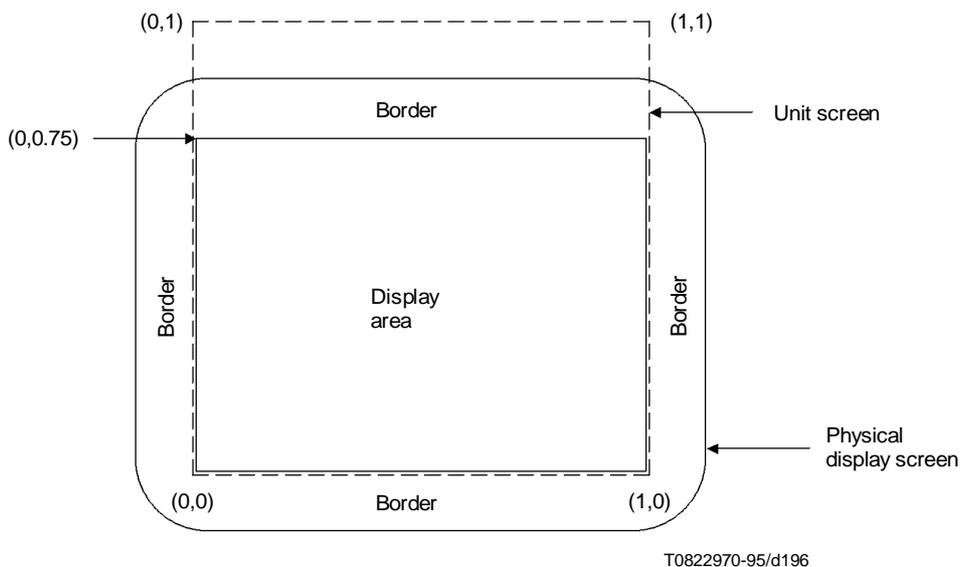
The coordinate system used in this Data Syntax is based on the abstract concept of a three-dimensional cartesian space with unit coordinates. This achieves independence of display hardware constraints. The coordinates are the width (X), height (Y), and depth (Z), and the range of possible values for each coordinate is from 0 (inclusive) to 1 (non-inclusive). Note that the Z coordinate is only meaningful on receiving devices with a capability for operating in three-dimensional mode. It is allowed in data structures both for logical completeness and to facilitate the graceful introduction of this feature when the technology becomes available. In this Data Syntax, two-dimensional mode will be assumed, with complete specification of the three-dimensional mode operation deferred for future standardization. In general, therefore, descriptions will deal with a two-dimensional (X, Y) plane of the space at  $Z = 0$ . This plane will be referred to as the unit screen. A value of  $Z = 0$  is interpreted as being farthest from the user.

Drawing of alphanumeric characters and pictorial information always occurs within the unit screen. The unit screen is visible in the display area, which is a rectangular area of the device's physical display screen. The lower left corner of the unit screen is the origin (0, 0), and coincides with the lower left corner of the display area. While the entire display area is always visible, the amount of the unit screen visible in the display area is implementation-dependent. Note that although it is always permissible to draw anywhere on the unit screen, only that portion of the unit screen coinciding with the display area is visible.

For example, on the cathode ray tubes used in television sets, the physical display screen usually has an aspect ratio of approximately 4:3 (width:height). If the display area on such a device has the same 4:3 aspect ratio, then the visible portion of the unit screen will be X from 0 (inclusive) to 1 (non-inclusive), and Y from 0 (inclusive) to approximately 0.75 (see Figure D.2).

The border area, if any, is not part of the display area, and no portion of the unit screen is visible in the border. Note that the physical display screen may also be used by other implementation-dependent display processes. processes are outside the scope of this Data Syntax.

NOTE – The sender can always assume that the entire display area is available. It is the responsibility of the receiving device to ensure that the entire display area is visible to the user.



**Figure D.2/T.101 – Unit screen concept**

### D.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 the effects of each superimposed over those of previous ones. In this manner, pictures are built up in layers. If a subsequent drawing command or alphabetic character affects a given pixel of the display, it supersedes the effect of any previous command on that pixel. 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 and the superimposition between these drawing modes shall be maintained to within one pixel, consistent with the physical resolution.

### D.4.3 Code Extension

#### D.4.3.1 General

**D.4.3.1.1** The method of code extension used in this Data Syntax is based on the code extension techniques specified in ISO 2022.

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.

Using the above code extension technique, any control or graphic set that is registered in accordance with ISO 2375 can be designated and invoked and used in conjunction with any or all of the control and graphic sets (including the PDI set) of this Data Syntax, within the context of the code environment defined by this Data Syntax. Specifically, using this technique, this Data Syntax can cater for other linguistic requirements such as Greek, Cyrillic, Chinese Hanzi, Japanese Kanji and Katakana, Arabic, etc.

**D.4.3.1.2** The entire coding environment described in this Data Syntax is to be designated and invoked as a 'complete code' by the escape sequence ESC 2/5 4/1, in accordance with ISO 2022. 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 invoked into columns 2 to 7, 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 4/0 or by the designation and invocation of any other “complete code”.

**D.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.

The registered Final Characters for all the G-sets (Graphic-Sets) and C-sets (Control-sets) are as given in Recommendation T.51.

**D.4.3.1.4** The choice of the 7-bit or 8-bit code environment may be explicitly established or changed for a particular service, or may be implicitly established 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.

#### **D.4.3.2 Code extension in a 7-bit environment**

An in-use table with 128 code positions is defined, as shown in Figure D.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 six columns of the in-use table. The manner in which this is accomplished is graphically depicted in Figure D.4 and is described below.

A single additional active control set, the C1-set, and four active G-sets, the G0, G1, G2, and G3 sets, are defined. The contents of the C1-set are described in D.6.2. The contents of the G0, G1, G2, and G3 sets can be dynamically selected from the larger repertory of G-sets 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).

ISO 2022-1982 specifies that the syntax of an escape sequence is ESC I...I F, where I...I is zero or more occurrences of intermediate characters in the range 2/0 through 2/15 and F is one occurrence of a final character in the range 3/0 through 7/14. The occurrence of any other bit combination in an escape sequence shall cause the partial escape sequence to be terminated and ignored, and that bit combination shall be executed.

The final character determines which set from the larger repertory is to be selected. Table D.1 shows the I and F character pairs assigned to each C- and G-set. The F character for the primary character set, for example, is 4/2 and for G0 the I character is 218. The three character escape sequence ESC 2/8 4/2, therefore, designates the primary character set as the current G0-set.

				b <sub>7</sub>	0	0	0	0	1	1	1	1
				b <sub>6</sub>	0	0	1	1	0	0	1	1
				b <sub>5</sub>	0	1	0	1	0	1	0	1
b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	Colonne Rangée	0	1	2	3	4	5	6	7
0	0	0	0	00	Jeu C0	Jeu G						SP
0	0	0	1	01								
0	0	1	0	02								
0	0	1	1	03								
0	1	0	0	04								
0	1	0	1	05								
0	1	1	0	06								
0	1	1	1	07								
1	0	0	0	08								
1	0	0	1	09								
1	0	1	0	10								
1	0	1	1	11								
1	1	0	0	12								
1	1	0	1	13								
1	1	1	0	14								
1	1	1	1	15								
												DEL

T0822980-95/d197

**Figure D.3/T.101 – 7-bit in-use table**

The designation and invocation sequences for the C0 and C1-sets are ESC 2/1 4/11 and ESC 2/2 4/6 respectively. All designation sequences designating G- and C1-sets defined in this Data Syntax shall be implemented. Other G- and C1-sets defined according to the rules of ISO 2022 may be implemented. All other designation sequences shall designate either a null G- or a null C1-set. The redesignation of the C0-set is not permitted in the context of this Data Syntax and such redesignating escape sequences shall be ignored. A null set is a set in which all code positions are executed as null operations.

The SHIFT-IN (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 SHIFT-OUT (SO) character is used to invoke the current G1-set into the in-use table in a locking manner. The sequence LOCKING-SHIFT TWO (LS2) is used to invoke the G2-set into the in-use table in a locking manner. The sequence LOCKING-SHIFT THREE (LS3) is used to invoke the G3-set into the in-use table in a locking manner. Table D.2 shows the coding of the shift functions.

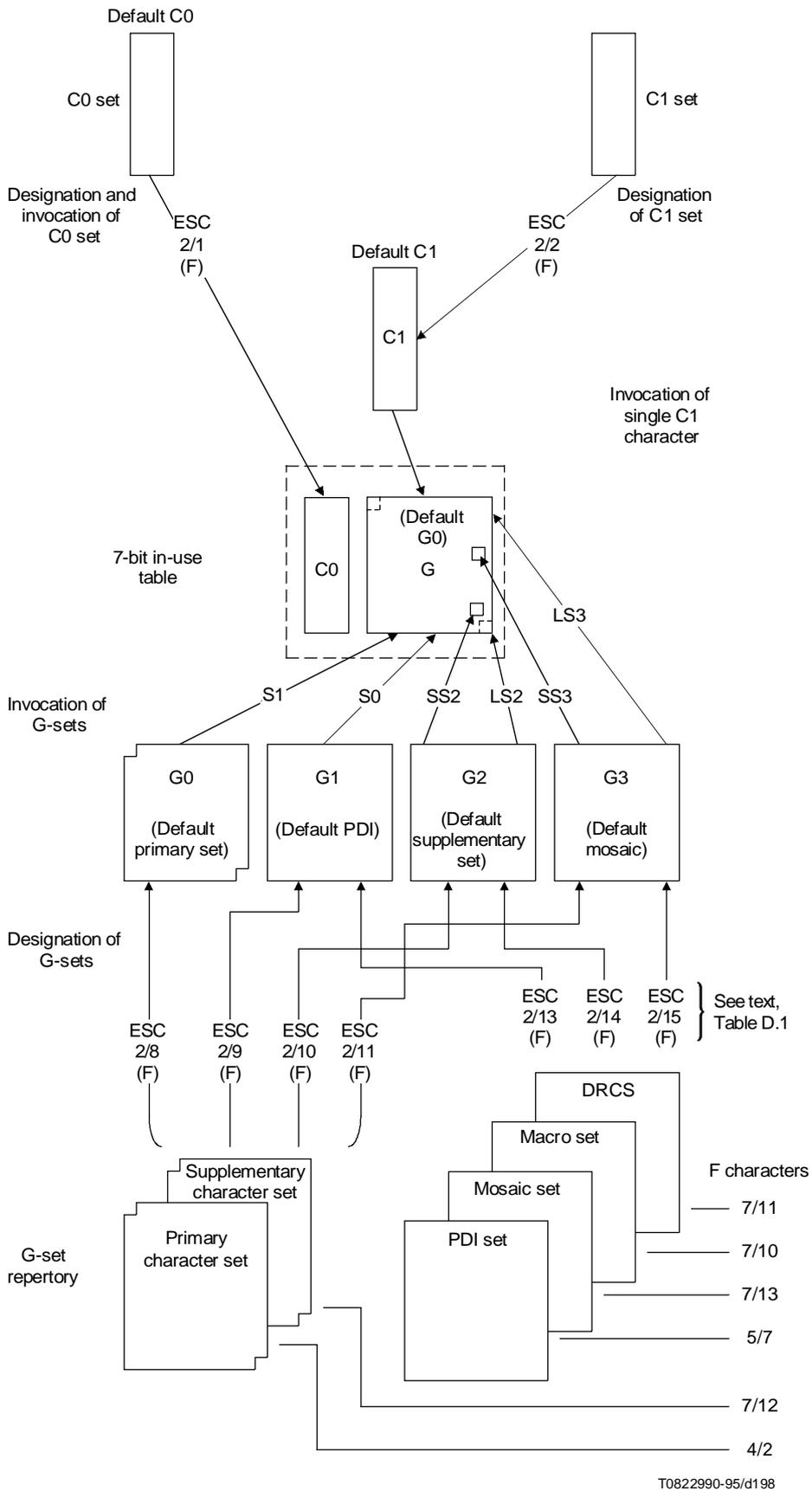


Figure D.4/T.101 – Code extension in a 7-bit environment

**Table D.1/T.101 – Escape sequences for designation of C- and G-sets**

Escape sequence	Set to be designated
Control sets: ESC 2/1 4/11 ESC 2/2 4/6  94-character sets: ESC I 4/2 ESC I 7/12  where I is 2/8, 2/9, 2/10, 2/11 for G0, G1, G2, G3 respectively  96-character sets: ESC I 5/7 ESC I 7/13 ESC I 7/10 ESC I 2/0 7/10 ESC I 7/11 ESC I 2/0 7/11  where I is 2/9, 2/10, 2/11 for G1, G2, G3 respectively <sup>a)</sup> I is also 2/13, 2/14, 2/15 for G1, G2, G3 respectively	C0-set C1-set  Primary character set Supplementary character set  PDI set Mosaic set Macro set <sup>b)</sup> DRCS set <sup>b)</sup>
<sup>a)</sup> There are two I characters that will result in the redesignation of each of the three G-sets, G1, G2 and G3. This dual coding may be removed from the Data Syntax in a future revision.  <sup>b)</sup> The 4-byte escape sequence is in accordance with ISO 2022 for DRCS. The Macro set is treated like the DRCS set. Future data base and terminal equipment should support double coding of both the 3-byte and 4-byte escape sequences.	

The single-shift characters, SINGLE-SHIFT TWO (SS2) and SINGLE-SHIFT THREE (SS3), are used to invoke, in a nonlocking 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. If a C0 character immediately follows an SS2 or SS3 character (instead of a byte from columns 2 to 7), the SS2 or SS3 character is ignored and the C0 character is executed. 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.

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 and represents the corresponding C1 character of column 8 or 9. 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 redesignated 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 D.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 D.5.3) is designated as the G1-set, the supplementary character set (see D.5.2) is designated as the G2-set, and the mosaic set (see D.5.4) is designated as the G3-set, all by default.

**Table D.2/T.101 – Coding of shift functions**

Shift function		7-bit environment	8-bit environment	G-set invoked
SHIFT-IN	SI	0/15	0/15	G0 into GL
SHIFT-OUT	SO	0/14	0/14	G1 into GL
LOCKING-SHIFT ONE RIGHT	LS1R	–	ESC 7/14 <sup>a)</sup>	G1 into GR
LOCKING-SHIFT TWO	LS2	ESC 6/14	ESC 6/14	G2 into GL
LOCKING-SHIFT TWO RIGHT	LS2R	–	ESC 7/13 <sup>a)</sup>	G2 into GR
LOCKING-SHIFT THREE	LS3	ESC 6/15	ESC 6/15	G3 into GL
LOCKING-SHIFT THREE RIGHT	LS3R	–	ESC 7/12 <sup>a)</sup>	G3 into GR
SINGLE-SHIFT TWO	SS2	1/9	1/9	G2 (nonlocking)
SINGLE-SHIFT THREE	SS3	1/13	1/13	G3 (nonlocking)

<sup>a)</sup> LS1R, LS2R, and LS3R are also coded as ESC 6/11, ESC 6/12, and ESC 6/13, respectively. This dual coding may be removed from this Data Syntax in a future revision.

### D.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. An in-use table with 256 code positions is defined, as shown in Figure D.5. Again, each incoming bit combination is either decoded according to the contents of this table or used to change the contents of this table. The table itself is organized into 16 columns of 16 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. The use of ESC Fe sequences to represent C1 characters (see D.4.3.2) is permitted, although not encouraged, in an 8-bit environment. 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 D.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 LOCKING-SHIFT ONE RIGHT (LS1R) is used to invoke, in a locking manner, the G1-set into GR. The escape sequences LOCKING-SHIFT TWO (LS2) and LOCKING-SHIFT TWO RIGHT (LS2R) are used to invoke, in a locking manner, the G2-set into GL and GR, respectively. The escape sequences LOCKING-SHIFT THREE (LS3) and LOCKING-SHIFT THREE RIGHT (LS3R) are used to invoke, in a locking manner, the G3-set into GL and GR, respectively. See Table D.2 for the coding of the shift functions.

Note also that the G2 and G3-sets can be invoked into GL in a non-locking manner using the SS2 and SS3 characters, respectively, as described for use in a 7-bit environment. If the byte immediately following the SS2 or SS3 character is from columns 10 to 15, bit  $b_8$  is ignored.

Upon initialization, the primary, PDI, supplementary, and mosaic sets are designated as G0, G1, G2 and G3-sets, respectively, and G0 is invoked into GL by default as in the 7-bit environments. In addition, G1 is invoked into GR.

				b <sub>8</sub>	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				b <sub>7</sub>	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				b <sub>6</sub>	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				b <sub>5</sub>	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	Col- Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	00	C0		SP													
0	0	0	1	01																
0	0	1	0	02																
0	0	1	1	03																
0	1	0	0	04																
0	1	0	1	05																
0	1	1	0	06																
0	1	1	1	07																
1	0	0	0	08																
1	0	0	1	09																
1	0	1	0	10																
1	0	1	1	11																
1	1	0	0	12																
1	1	0	1	13																
1	1	1	0	14																
1	1	1	1	15																

T0823000-95/d199

Figure D.5/T.101 – 8-bit in-use table

**D.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 G-set with 94 code positions is invoked.

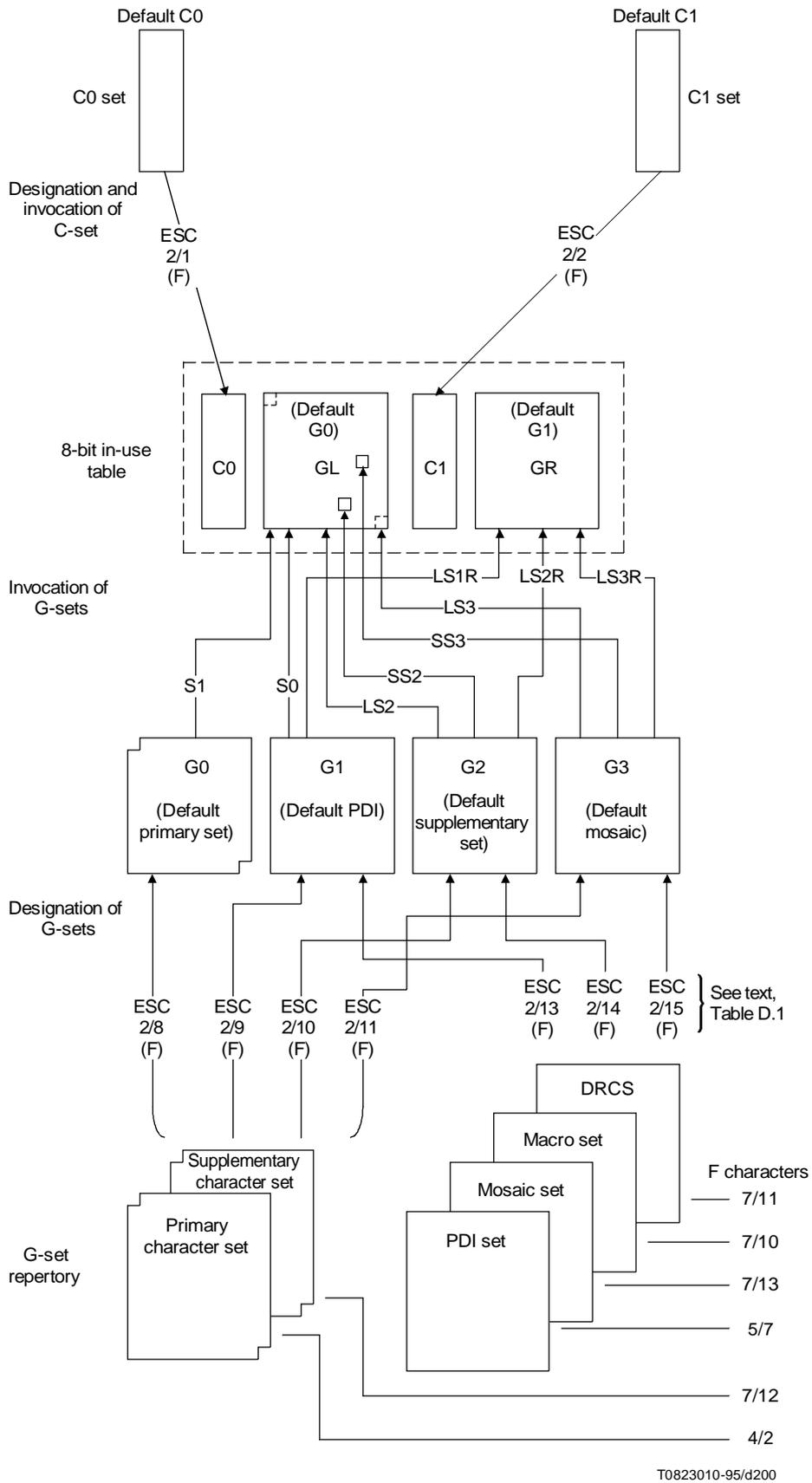
DELETE has been used primarily to erase or obliterate erroneous or unwanted characters in punched tape. The coding of DELETE is 7/15 in the 7-bit and 8-bit in-use table when a G-set with 94 code positions is invoked. DELETE is executed as a null operation in this annex.

**D.5 Coding of G-sets**

**D.5.1 Primary character set**

The primary character set consists of 94 Latin alphabetic characters, digits, punctuation marks, and symbols as illustrated in Figure D.7. This set is identical to the CCITT Recommendation T.50 (former Recommendation V.3, 1980), national version adopted by Canada and the United States of America. The particular patterns (font) chosen for the characters are implementation-dependent 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 colours, and at all display resolutions. When any character of the primary set is received, the cursor is automatically advanced (see D.5.3.2.3.4).

The sequence used to designate the primary character set is ESC I 4/2, where I is 2/8, 2/9, 2/10, or 2/11 to indicate G0, G1, G2 or G3, respectively (see D.4.3).



T0823010-95/d200

Figure D.6/T.101 – Code extension in an 8-bit environment

					10	11	12	13	14	15	
					b <sub>7</sub>	0	0	1	1	1	1
					b <sub>6</sub>	1	1	0	0	1	1
					b <sub>5</sub>	0	1	0	1	0	1
b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	Column Row	2	3	4	5	6	7	
0	0	0	0	00		0	@	P	'	p	
0	0	0	1	01	!	1	A	Q	a	q	
0	0	1	0	02	"	2	B	R	b	r	
0	0	1	1	03	#	3	C	S	c	s	
0	1	0	0	04	\$	4	D	T	d	t	
0	1	0	1	05	%	5	E	U	e	u	
0	1	1	0	06	&	6	F	V	f	v	
0	1	1	1	07	'	7	G	W	g	w	
1	0	0	0	08	(	8	H	X	h	x	
1	0	0	1	09	)	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		

T0823020-95/d201

**Figure D.7/T.101 – Primary character set**

## D.5.2 Supplementary character set

The supplementary character set of accents, diacritical marks, and special characters for Latin-based alphabets is illustrated in Figure D.8. This table is based on CCITT Recommendation S.100 (1980) and includes some additional characters proposed by CCIR, 150, and other organisations. The particular patterns (font) chosen for the characters are implementation-dependent 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 D.5.3.2.3.4.

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. Only certain combinations of non-spacing characters of the supplementary set are combined with the characters of the primary set to form characters of the graphic character repertoire (see D.7.2). 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 sequence used to designate the supplementary character set is ESC I 7/12, where I is 2/8, 2/9 2/10, or 2/11 to indicate G0, G1, G2, or G3, respectively (see D.4.3).

					10	11	12	13	14	15	
					b <sub>7</sub>	0	0	1	1	1	1
					b <sub>6</sub>	1	1	0	0	1	1
					b <sub>5</sub>	0	1	0	1	0	1
b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	Column Row	2	3	4	5	6	7	
0	0	0	0	00		0	—	—	Ω	κ	
0	0	0	1	01	ı	±	`	ˆ	Æ	æ	
0	0	1	0	02	ç	²	´	®	Ð	đ	
0	0	1	1	03	£	³	^	©	à	ð	
0	1	0	0	04	\$	×	~	T.M.	Ĥ	ĥ	
0	1	0	1	05	¥	μ	—	♪	⊞	ı	
0	1	1	0	06	#	¶	˘	☐	ıı	ij	
0	1	1	1	07	§	.	·	☐	L˘	ı˘	
1	0	0	0	08	¤	÷	..	☐	Ł	ł	
1	0	0	1	09	‘	’	/	☐	Ø	ø	
1	0	1	0	10	“	”	◦	◼	Œ	œ	
1	0	1	1	11	«	»	¸	◼	◌̊	β	
1	1	0	0	12	←	¼	☐	⅛	þ	ƒ	
1	1	0	1	13	↑	½	"	⅜	ƒ	ƒ	
1	1	1	0	14	→	¾	¸	⅝	ŋ	ŋ	
1	1	1	1	15	↓	¿	∨	⅞	‘n		

T0823030-95/d202

NOTE – Column 4 (12) is non-spacing. Also, the rectangles surrounding the characters in 4/12, 5/6 through 5/11, and 6/5 are for illustrative purposes only and are not part of the graphic symbols.

Figure D.8/T.101 – Supplementary character set

## D.5.3 Picture Description Instruction (PDI) set

### D.5.3.1 General

**D.5.3.1.1** The Picture Description Instruction (PDI) set, shown in Figure D.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 COLOUR, WAIT, SELECT COLOUR, 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 predefined patterns, one per character, but executable drawing functions that produce an image not necessarily restricted to a single character field.

The sequence used to designate the PDI-set is ESC I 5/7, where I is 2/9 or 2/13 to indicate G1, 2/10 or 2/14 to indicate G2, or 2/11 or 2/15 to indicate G3 (see D.4.3).

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  $b_7$  is set to 0, an opcode is indicated. If  $b_7$  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. The transmission control characters (0/1-0/6, 1/0, 1/5-1/7), the device control characters (1/1-1/4), and NULL (0/0) have no effect on the presentation layer and, therefore, do not terminate PDI sequences (see D.6.1.4, D.6.1.5, and D.6.1.6.1). The invocation of a macro either from the in-use table or by single shift will not by itself terminate a PDI; the PDI may be continued in the operand data contained in the macro. 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.  $b_6$  to  $b_1$ . The single-value operands consist of one to four bytes of numeric data, as determined by the DOMAIN command described in D.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  $b_6$  to low order bit or  $b_1$ ) from the numeric data bytes as shown in Figure D.10.

The multi-value operands consist of one to 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 colour information (when used in conjunction with the SET COLOUR 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 coordinates 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 D.5.3.3.

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

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. The precision to which the position of the cursor and drawing point must be maintained is implementation-dependent and shall be consistent with the physical resolution. 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. The handling of this error condition is implementation-dependent. For example, this PDI may either be rejected (i.e. executed as a null operation) or executed and clipped within the unit screen.

					10	11	12	13	14	15	
					b <sub>7</sub>	0	0	1	1	1	1
					b <sub>6</sub>	1	1	0	0	1	1
					b <sub>5</sub>	0	1	0	1	0	1
b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	Column Row	2	3	4	5	6	7	
0	0	0	0	00	C O N T R O L	R E C T A N G L E	Numeric data				
0	0	0	1	01							
0	0	1	0	02							
0	0	1	1	03							
0	1	0	0	04	P O I N T	P O L Y G O N					
0	1	0	1	05							
0	1	1	0	06							
0	1	1	1	07							
1	0	0	0	08	L I N E	I N C R E M E N T A L					
1	0	0	1	09							
1	0	1	0	10							
1	0	1	1	11							
1	1	0	0	12	A R C	C O N T R O L					
1	1	0	1	13							
1	1	1	0	14							
1	1	1	1	15							

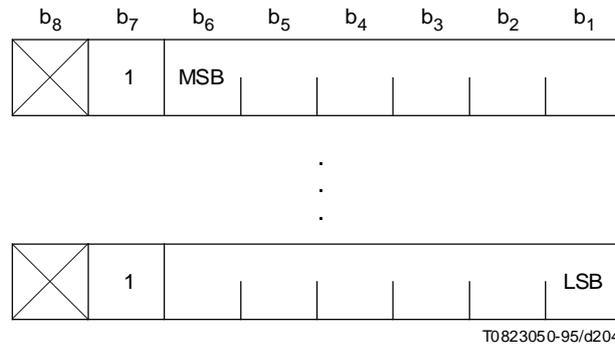
T0823040-95/d203

Figure D.9/T.101 – General PDI set

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

Each byte contains two three-tuples. Each three-tuple contains one bit for each of the three primary colours. These are specified in the order green, red, blue, which is the order of decreasing luminance. A complete colour 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 colour value thereby obtained represents a binary fraction in which the MSB acts as the digit just to the right of the decimal point.

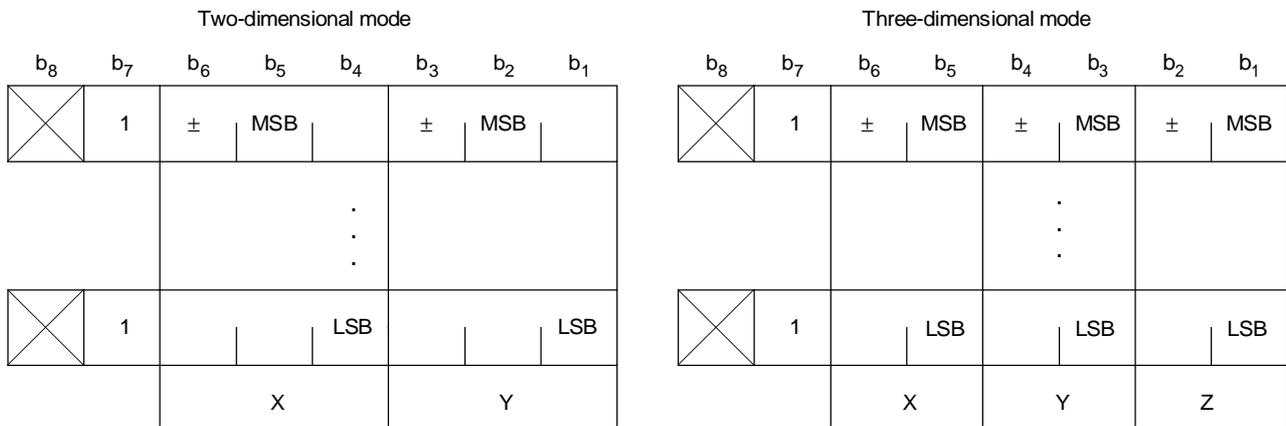
Table D.3 shows the types of operands used by each of the opcodes.



T0823050-95/d204

MSB Most significant bit of operand  
 LSB Least significant bit of operand. Most significant byte transmitted first

**Figure D.10/T.101 – Single-value format**



T0823060-95/d205

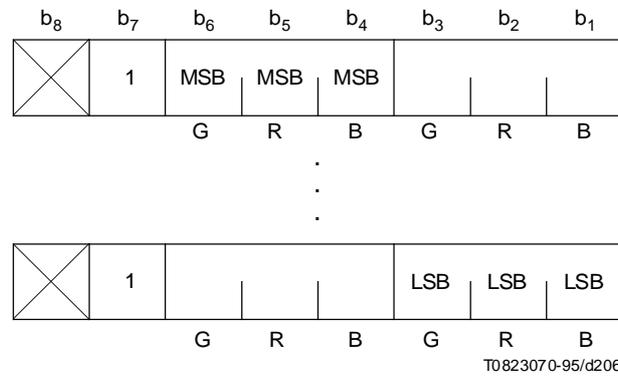
**Figure D.11/T.101 – Multi-value format**

**D.5.3.1.2** The functions of the opcodes are summarized as follows:

- 1) POINT sets the drawing point to any position in the unit screen and optionally displays a dot.
- 2) LINE draws a line based on its end points.
- 3) ARC draws a circular arc based on the end points of the arc and a point on the arc. The end points 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 end points coincide and whose intermediate point (with the end points) defines the diameter.
- 4) RECTANGLE draws a rectangular outline or fills in an area of specified length and width.
- 5) POLYGON draws a polygonal outline or fills in the circumscribed area based on a series of defined vertices.

- 6) INCREMENTAL draws a point, line, or polygon in an incremental manner.
- 7) CONTROL provides control over the modes of the drawing commands. One of its major functions is to set up a value or colour of an object

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



**Figure D.12/T.101 – Color value format**

**Table D.3/T.101 – Operand Types**

Opcode	Operand
RESET	Fixed
DOMAIN	Fixed/multi-value
TEXT	Fixed/multi-value
TEXTURE	Fixed/multi-value
SET COLOUR	Multi-value
WAIT	Fixed
SELECT COLOUR	Single-value
BLINK	Fixed/single-value
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

### D.5.3.2 Attribute control functions

#### D.5.3.2.1 General

The opcodes described in D.5.3.2.2 to D.5.3.2.9 control the attributes and display parameters.

					10	11	12	13	14	15	
					b <sub>7</sub>	0	0	1	1	1	1
					b <sub>6</sub>	1	1	0	0	1	1
					b <sub>5</sub>	0	1	0	1	0	1
b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	Column Row	2	3	4	5	6	7	
0	0	0	0	00	Reset	Rect. (out-lined)	Numeric data				
0	0	0	1	01	Domain	Rect. (filled)					
0	0	1	0	02	Text	Set a rect. (outlined)					
0	0	1	1	03	Texture	Set a rect. (filled)					
0	1	0	0	04	Point set (abs.)	Poly. (out-lined)					
0	1	0	1	05	Point set (rel.)	Poly. (filled)					
0	1	1	0	06	Point (abs.)	Set a poly. (out-lined)					
0	1	1	1	07	Point (rel.)	Set a poly. (filled)					
1	0	0	0	08	Line (abs.)	Field					
1	0	0	1	09	Line (rel.)	Incr. point					
1	0	1	0	10	Set a line (abs.)	Incr. line					
1	0	1	1	11	Set a line (rel.)	Incr. poly. (filled)					
1	1	0	0	12	Arc (out-lined)	Set colour					
1	1	0	1	13	Arc (filled)	Wait					
1	1	1	0	14	Set an arc (out-lined)	Select colour					
1	1	1	1	15	Set an arc (filled)	Blink					

T0823080-95/d207

Figure D.13/T.101 – PDI set

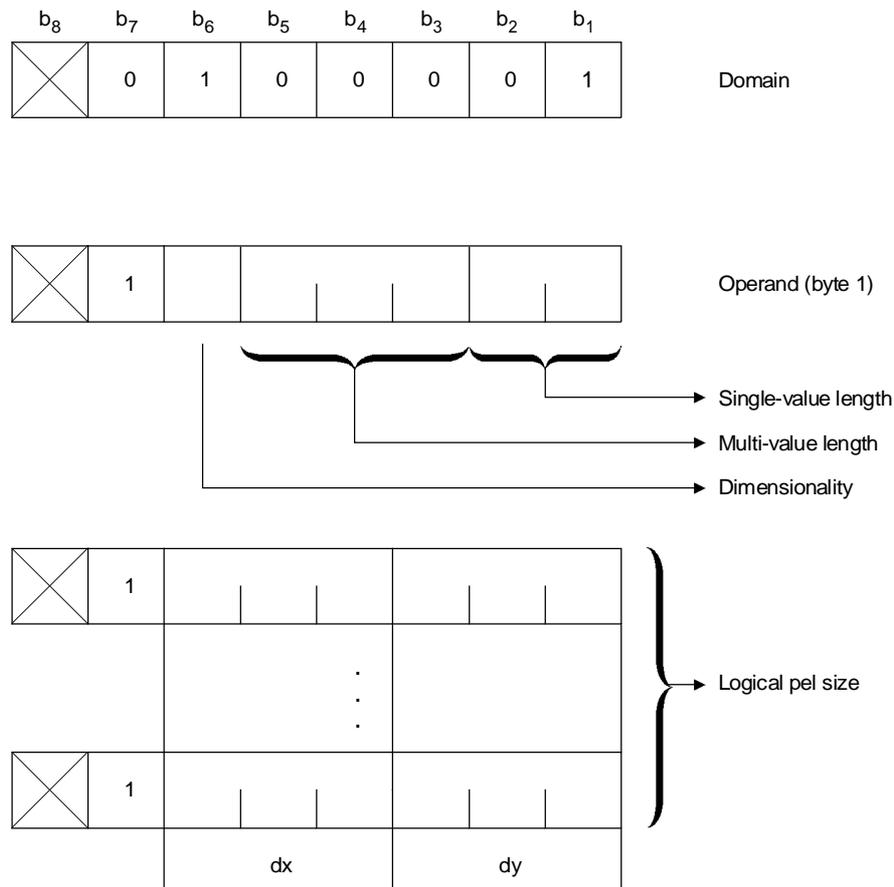
## D.5.3.2.2 DOMAIN

### D.5.3.2.2.1 Command format

The DOMAIN command is used to control the precision of single-value and multi-value operands, the dimensionality of coordinate specifications, and the size of the logical pel. (See Figure D.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 D.6.1.6.5. The DOMAIN opcode takes a one byte, fixed format operand, followed by a multi-value operand, whose interpretations are shown below.

#### D.5.3.2.2.2 Single-value operand length

Bits b<sub>2</sub> and b<sub>1</sub> of byte 1 determine the length, that is, the number of bytes to be used in single-value operands, as shown in Table D.4. The default length is one byte.



T0823090-95/d208

Figure D.14/T.101 – Domain

Table D.4/T.101 – Single-value operand length

$b_2$	$b_1$	Number of bytes
0	0	1 (default value)
0	1	2
1	0	3
1	1	4

#### D.5.3.2.2.3 Multi-value operand length

Bits  $b_5$ ,  $b_4$ , and  $b_3$  of byte 1 determine the length, that is, the number of bytes to be used in multi-value operands, as shown in Table D.5. The default length is three bytes.

#### D.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 coordinates 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.

**Table D.5/T.101 – Multi-value operand length**

b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	Number of 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

**D.5.3.2.2.5 Operand 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), trailing zero bits are supplied by the receiving presentation process, 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 annex.

**D.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 coordinates system. This multi-value operand specifies the logical pel size to be used with the POINT, LINE, ARC, RECTANGLE, POLYGON, and INCREMENTAL PDI'S, as well as UNDERLINE START, separated mosaics, line textures, and texture patterns, but not for the display of alphanumeric characters (including underline and non-spacing underline). 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, with the origin at the lower left.

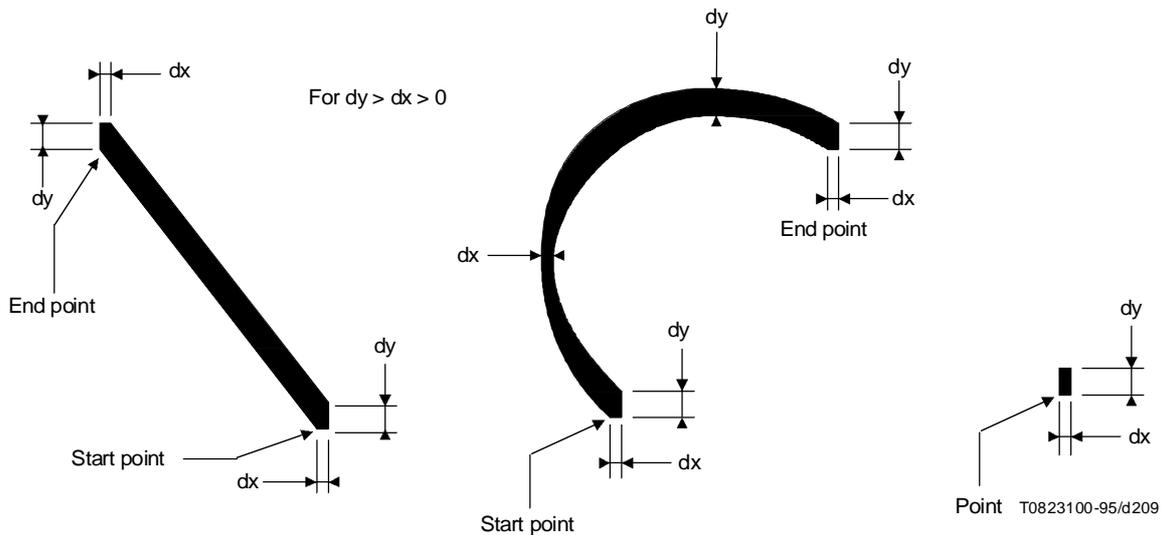
A drawing primitive is defined by an implementation dependent algorithm that describes as closely as possible a precise geometric path for all displacements, including zero. For example, a LINE is a locus of points following a straight line algorithm between two specified coordinates. The physical picture 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 D.15, which is illustrative of the application of logical pel to a LINE, an ARC, and a POINT.

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

- 1) Lower left corner if both dx and dy are positive.
- 2) Lower right corner if dx is negative and dy is positive.
- 3) Upper left corner if dx is positive and dy is negative.
- 4) 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.

Additional numeric data bytes following the logical pel size data byte(s) are reserved for future standardization and shall be ignored. If the logical pel size operand is omitted, the size of the logical pel shall not be changed.



**Figure D.15/T.101 – Application of logical pel to a LINE, an ARC, and a POINT**  
(Effect of logical pel size on stroke width)

### D.5.3.2.3 TEXT

#### D.5.3.2.3.1 Command format

This command 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 D.16.)

#### D.5.3.2.3.2 Character rotation

Bits  $b_2$  and  $b_1$  of byte 1 are used to specify character rotation as shown in Table D.6.

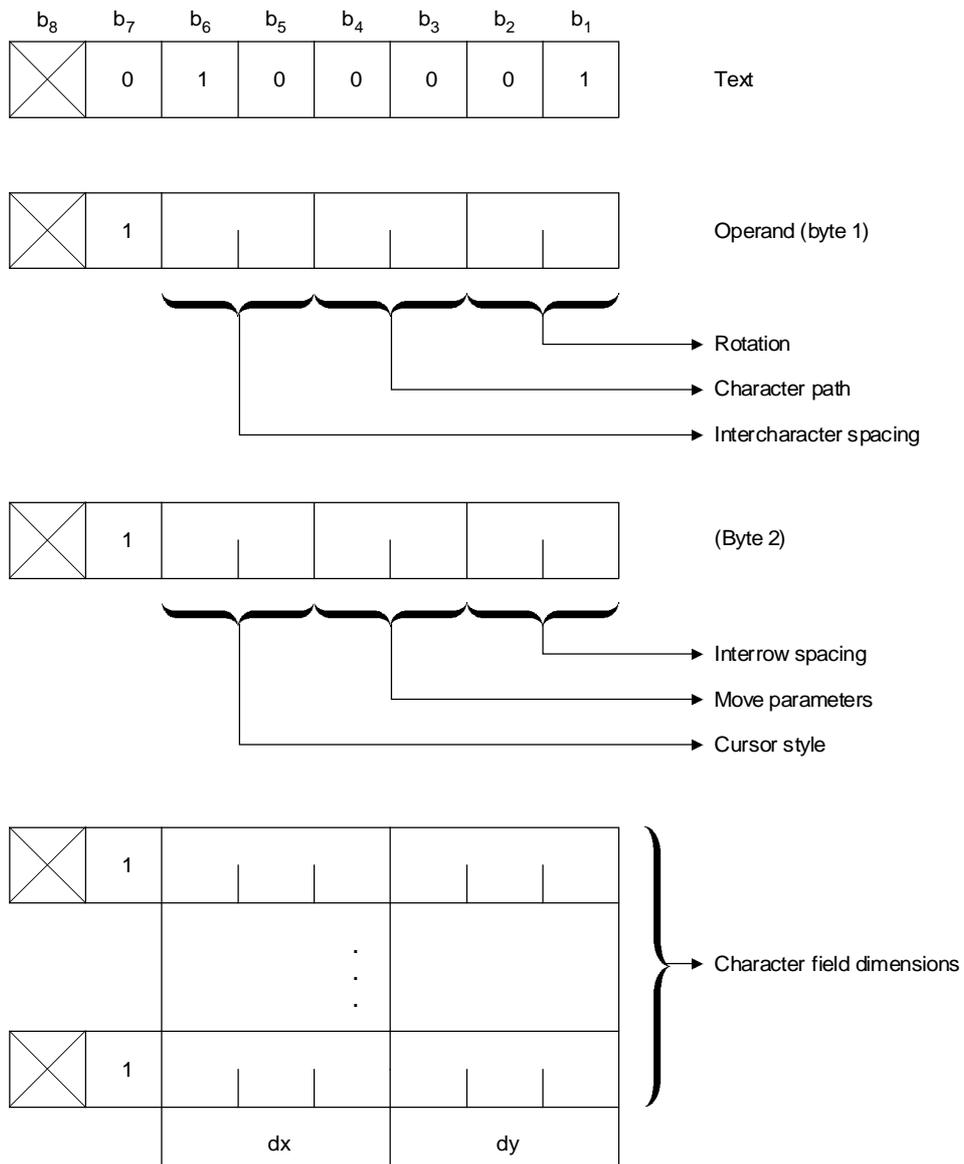
Rotation causes the character field and the cursor to rotate counterclockwise about the character field origin. This rotation is measured relative to horizontal within the unit screen and is independent of the character path. The character field origin is the lower left corner of the character field at the default 0 degree rotation regardless of the sign of the character field dimensions  $dx$  and  $dy$  (see Figure D.17). All alphanumeric characters (including diacritical marks and underlines), DRCS, mosaics, and separated mosaic characters, and the underline produced when underline mode (see D.6.2.7.15) is in effect, are affected by rotation so that the relative position of the images within the character field is unchanged.

#### D.5.3.2.3.3 Character path movement

Bits  $b_4$  and  $b_3$  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 D.7 describes the four possible character paths. The character path is defined relative to horizontal within the unit screen and is independent of the character rotation. The default character path is right.

#### D.5.3.2.3.4 Intercharacter spacing

Bits  $b_6$  and  $b_5$  of byte 1 are used to determine the distance the cursor is moved after a character is displayed or after a SPACE or APB (backspace) or APF (horizontal tab) character is received. The distance the cursor is moved is in multiples of the character field width ( $dx$ ) or height ( $dy$ ), whichever lies parallel to the character path, depending on the character path and character rotation. This is known as the intercharacter spacing and is as defined in Table D.8.

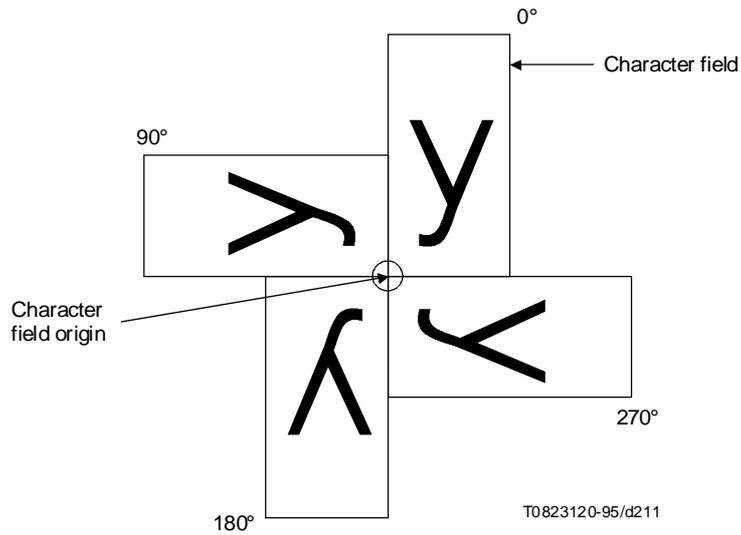


T0823110-95/d210

**Figure D.16/T.101 – TEXT**

**Table D.6/T.101 – Character rotation**

$b_2$	$b_1$	Degrees of rotation
0	0	0 (default value)
0	1	90
1	0	180
1	1	270



**Figure D.17/T.101 – Character rotation**

**Table D.7/T.101 – Character path**

b <sub>4</sub>	b <sub>3</sub>	Cursor movement
0	0	Right (default value)
0	1	Left
1	0	Up
1	1	Down

**Table D.8/T.101 – Intercharacter spacing**

b <sub>6</sub>	b <sub>5</sub>	Spacing
0	0	1 (default value)
0	1	5/4
1	0	3/2
1	1	Proportional spacing

The three fixed intercharacter spacings (1, 5/4, and 3/2, consistent with the physical resolution) are interpreted as multiplicative functions of the dimension of the current character field lying parallel to the character path that are applied to movements of the cursor. In the proportionally spaced mode, the intercharacter 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 implementation dependent. However, each character shall be completely contained within the area defined by the current character field (see D.5.3.2.3.9). This means that the exact number of characters per line is not known in proportional spacing mode, but it is at least as many characters per line as would be allowed by the current character field dimensions. Note that in order to guarantee the display of proportionally spaced text within an active field (see D.5.3.3.6.2) on all implementations, without unintentional wrap (see D.6.2.7.11 and D.5.3.2.3.6) or scroll (see D.6.2.7.13), either:

- 1) the field should be large enough to wholly contain the text as though the text were not proportionally spaced; or
- 2) the number and size of the characters should be small enough to fit within the field as though the text were not proportionally spaced.

The width of the character field does not change. For example, when a character is displayed at the end of a line in colour mode 2, the background colour is displayed for the full width of the character field. The default intercharacter spacing is a fixed space of 1 in which the current character field abuts the previous character field.

#### D.5.3.2.3.5 Interrow spacing

Bits  $b_2$  and  $b_1$  of byte 2 determine the interrow spacing of characters, which defines the relative location of the cursor when it is advanced to a new line in a direction perpendicular ( $-90$  degrees) to the character path, either automatically as described below or by the APD (line feed) or APU (vertical tab) characters, as defined in D.6.1.2. Table D.9 shows the four interrow spacings (1,  $5/4$ ,  $3/2$ , and 2, consistent with the physical resolution), which are interpreted as multiples of the character field width (dx) or height (dy), whichever lies perpendicular to the character path, depending on the character path and character rotation. An interrow spacing of 1, in which the character field on the current row abuts the character field of the previous row, is the default.

**Table D.9/T.101 – Interrow spacing**

$b_2$	$b_1$	Spacing
0	0	1 (default value)
0	1	$5/4$
1	0	$3/2$
1	1	2

#### D.5.3.2.3.6 Automatic APR APD

When using fixed or proportional intercharacter spacing, if the result of moving the cursor (with a format effector or as a result of displaying a character) would cause any part of the full corresponding character field to be outside of the unit screen (or outside of the active field – see D.5.3.3.6.2, FIELD – if the character field was entirely within the active field immediately before the movement), an automatic APR (carriage return) and APD (line feed) are immediately executed. If an explicit APR APD (or APD APR) sequence is received after an automatic APR APD is executed but before the character field origin is moved, aligned, or set by any other received command or sequence, the explicit APR APD (or APD APR) sequence shall be executed as a null operation.

#### D.5.3.2.3.7 Move attributes

Bits  $b_4$  and  $b_3$  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 D.10.

**Table D.10/T.101 – Move attributes**

b <sub>4</sub>	b <sub>3</sub>	Attribute
0	0	Move together (default value)
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.

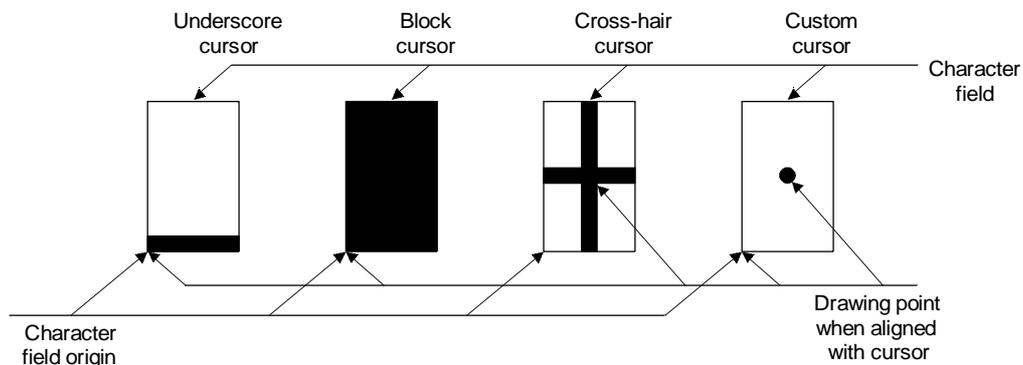
If the cursor is defined as leading (01), then every time the cursor is moved, the drawing point will move along with it, but not vice versa.

If the drawing point is set to lead the cursor (10), 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 shall be adjusted as close as possible to the drawing point without violating the above condition. Subsequent relative cursor positioning operations shall be made with reference to the adjusted cursor position.

The alignment of the drawing point corresponds to the character field origin for the underscore cursor and block cursor, and the centre of the character field for the cross-hair cursor and custom cursor. (See Figure D.18.)



T0823130-95/d212

NOTE – The boxes around the cursors are for illustrative purposes only and are not part of the cursor style.

**Figure D.18/T.101 – Cursor styles**

The execution of a TEXT command shall cause alignment of the drawing point if 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, except if it would cause any part of the character field to fall outside the unit screen. In this case, the cursor shall be adjusted as described above.

#### D.5.3.2.3.8 Cursor Styles

Bits  $b_6$  and  $b_5$  of byte 2 are used to determine the display style of the cursor symbol as in Table D.11 and Figure D.18.

**Table D.11/T.101 – Cursor styles**

$b_6$	$b_5$	Style
0	0	Underscore (default value)
0	1	Block
1	0	Cross-hair
1	1	Custom

The cursor indicates the position at which the next character is to be displayed. The underscore cursor symbol is a single line the width of the current character field at the bottom of the character field. The block cursor symbol is a solid block whose size is the size of the current character field. The cross-hair cursor symbol consists of a vertical line and a horizontal line that intersect at the centre of the character field and whose height and width are equal to the height and width of the current character field. The thickness of the underscore cursor and that of the cross-hair cursor and the definition of the shape of the custom cursor symbol are implementation-dependent.

#### D.5.3.2.3.9 Character field dimensions

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

If dx is negative, the character patterns are reflected about the vertical centre axis of the character field. If dy is negative, the character patterns are reflected about the horizontal centre 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  $dx = 1/40$  and  $dy = 5/128$ , consistent with the physical resolution.

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

Additional numeric data bytes following the multi-value operand are reserved for future standardization and shall be ignored.

### D.5.3.2.4 TEXTURE

#### D.5.3.2.4.1 Command format

This command 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. The TEXTURE opcode takes a onebyte, fixed format operand, followed by a multi-value operand whose interpretations are shown below (see Figure D.19).

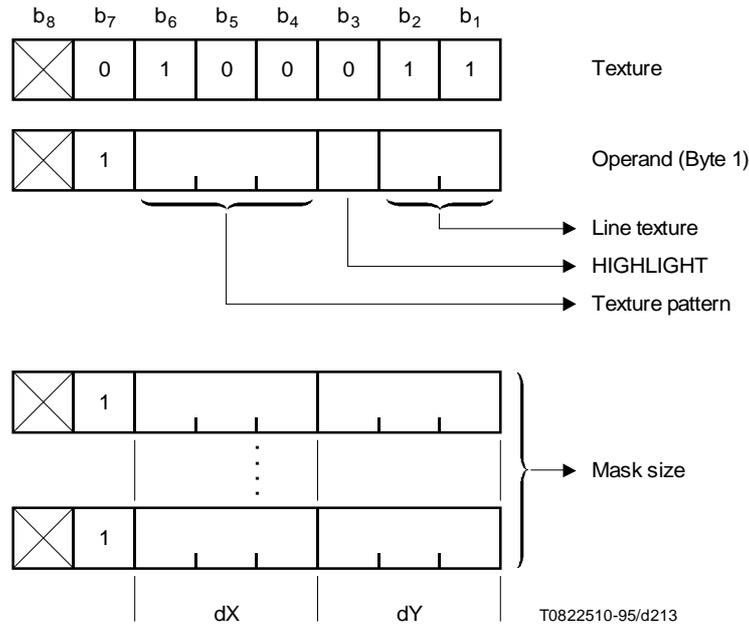


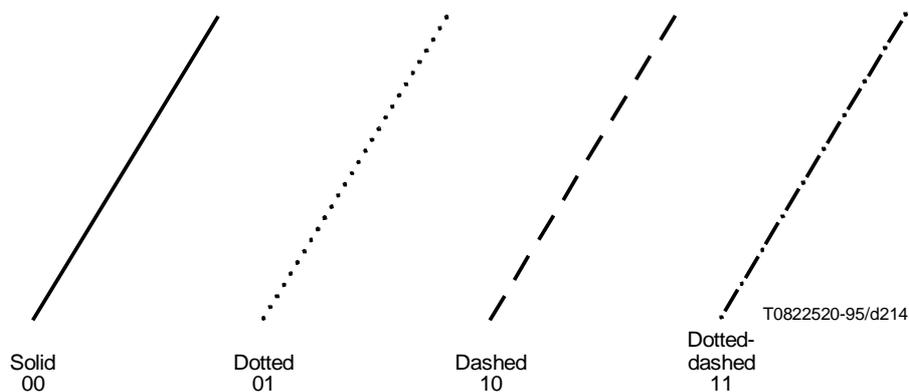
Figure D.19/T.101 – TEXTURE

#### D.5.3.2.4.2 Line texture

Bits  $b_2$  and  $b_1$  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 PDI's (see Figure D.20 and Table D.12). 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. In colour mode 2, the inter-dot and inter-dash spacing is drawn in the background colour.

The algorithm for generating line textures is implementation-dependent, and it should produce characteristics for arbitrary lines that yield a visually consistent effect with that specified for horizontal and vertical lines, although exact alignment is not guaranteed. All end points of lines and arcs and all vertices of incremental lines (with the draw flag on), highlighted incremental polygons, outlined or highlighted rectangles and polygons must be plotted regardless of the line texture used.

NOTE – If logical pel size  $dx = 0$ , all non-vertical lines are solid. If logical pel size  $dy = 0$ , all non-horizontal lines are solid.



**Figure D.20/T.101 – Line textures**

**Table D.12/T.101 – Line texture**

b <sub>2</sub>	b <sub>1</sub>	Texture
0	0	Solid (default value)
0	1	Dotted
1	0	Dashed
1	1	Dotted-dashed

#### D.5.3.2.4.3 Highlight

Bit b<sub>3</sub> 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, the line(s) or arc comprising the outline are drawn with solid line texture (independent of the current line texture) using the current logical pel size in nominal black in colour modes 0 and 1, and in the background colour in colour mode 2. The outline is the region traced out by the logical pel when the arc, rectangle, polygon, or incremental polygon is drawn. The default state of this attribute is no highlight (b<sub>3</sub> = 0). (See D.5.3.2.6 for a description of the three colour modes.)

#### D.5.3.2.4.4 Texture pattern

Bits b<sub>6</sub>, b<sub>5</sub>, and b<sub>4</sub> are used to select the texture pattern to be used in filling rectangles, arcs, polygons, and incremental polygons according to Table D.13 and Figure D.21.

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 shall be maintained across figures if the logical pel size is the same. For the predefined texture patterns, if the logical pel size is (0,0), solid texture patterns will always be drawn. In colour mode 2, the fill areas not drawn in the drawing colour are drawn in the background colour.

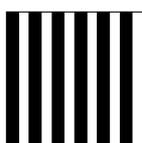
The programmable texture masks A, B, C and D are defined using the DEF TEXTURE command. The default pattern for the four programmable texture masks is a null texture pattern, resulting in no fill in colour modes 0 and 1, and a fill with the background colour in colour mode 2.

**Table D.13/T.101 – Texture pattern**

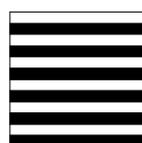
b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	Pattern
0	0	0	Solid (default value)
0	0	1	Vertical hatching
0	1	0	Horizontal hatching
0	1	1	Vertical and horizontal cross-hatching
1	0	0	Mask A
1	0	1	Mask B
1	1	0	Mask C
1	1	1	Mask D



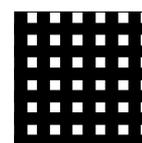
Solid



Vertical hatching



Horizontal hatching



T0822530-95/d215

Cross-hatching

**Figure D.21/T.101 – Texture patterns**

#### D.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 colour(s) in all pixels indicated by the mask pattern. This process takes as its reference the origin (0,0) point of the unit screen in order that registration of the pattern be maintained across figures at any given mask size.

The default mask size is dx = 1/40 and dy = 5/128, consistent with the physical resolution (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. Additional numeric data bytes following the mask size operand are reserved for future standardization and shall be ignored.

#### D.5.3.2.5 SET COLOUR

**D.5.3.2.5.1** The SET COLOUR command is used to specify colour values applied to all subsequent drawing commands and characters from the primary, supplementary, DRCS, and mosaic sets. It can also affect colours previously displayed. Three different colour modes can be selected, the choice of which dictates the precise interpretation of the two colour control opcodes, SET COLOUR and SELECT COLOUR. The colour mode is established to 0, 1 or 2 via the SELECT COLOUR PDI as described in D.5.3.2.6. Colour mode 0 is designed to support those situations in which the

drawing colour is directly specified as a colour value. Colours are implicitly defined in the colour map in this mode. Colour modes 1 and 2 are designed to make explicit use of a colour map capability. That is, the drawing colour is specified as an ordinal number that is used as an address into a look-up table that provides the actual colour value.

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

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

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

Colour mode 0 also uses the colour map. If the colour specified in the colour mode 0 SET COLOUR command has already been specified in the colour map, then the address of the drawing colour is set to the lowest address containing that colour and the colour map is not changed. If that colour has not already been specified in the colour map, colour mode 0 makes use of the lowest address that has not been used (via a colour mode 0, 1 or 2 SET COLOUR command or a colour mode 1 or 2 SELECT COLOUR command) since the last RESET command that reestablishes the default colour map, and that is not the address of nominal black or nominal white. If no addresses are available, then the colour map shall not be changed and the drawing colour is established in an implementation-dependent manner.

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

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

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

In colour mode 0, this sets the drawing colour. This drawing colour is applied to subsequently received alphanumeric and pictorial drawings until changed by another SET COLOUR command, the RESET command, described in D.5.3.2.9, or the NSR control character, described in D.6.1.6.5. The default drawing colour in colour mode 0 is white. A background colour cannot be specified in colour mode 0, that is, character patterns and pictorial drawings merely overwrite the existing contents of the physical display area which, otherwise, remain unchanged.

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

If the maximum size entry (i.e. number of bits) that the colour map can accommodate is smaller than the number of bits provided by the SET COLOUR operand, the operand is truncated and only the most significant bits are used. If the maximum size entry that the colour map can accommodate is larger than the number of bits provided by the SET COLOUR operand, trailing zero bits are supplied by the receiving presentation process. For each primary, the maximum colour fraction attainable, given the number of bits specified in the colour value operand, shall be interpreted as full intensity and intermediate values shall be equally distributed between zero and full intensity.

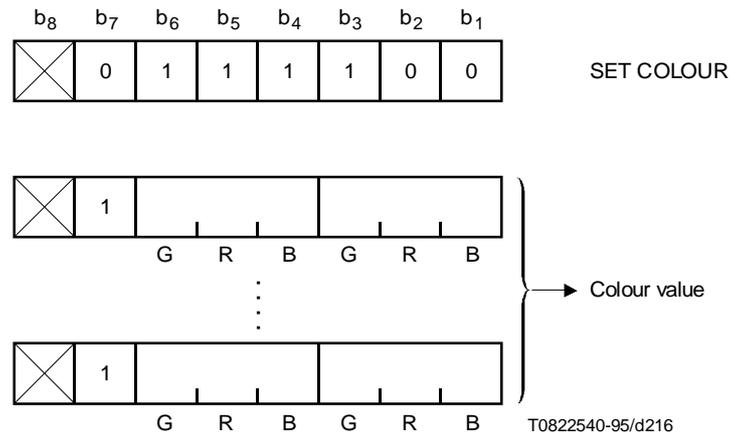


Figure D.22/T.101 – SET COLOUR

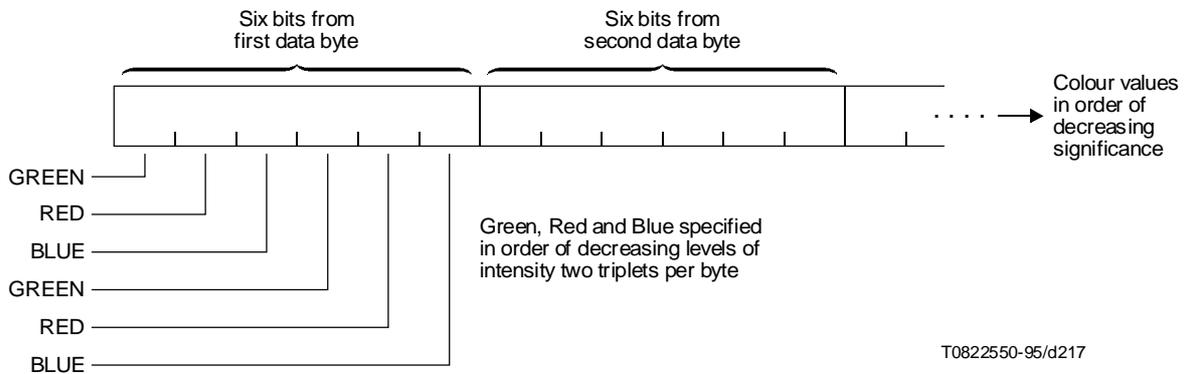


Figure D.23/T.101 – Colour encoding scheme

If the SET COLOUR command is implicitly repeated via the sending of additional numeric data, the address of the colour entry to be changed is automatically incremented prior to the execution of the new opcode. The algorithm for incrementing is to change the most significant zero to a one and to change all ones to the left of it to zero. For example, colour map address 010100 would be incremented to 110100, which in turn would be incremented to 001100. This incrementing does not affect the colour map address associated with the drawing colour. This incrementing process stops and subsequent operand data are ignored when the physical limit (all ones) of the implemented colour map is reached.

NOTE – This incrementing algorithm permits the same presentation to occur on a device with a greater number of colour map entries than the sender has assumed. Furthermore, careful placement of similar colours in adjacent colour map entries will permit a reasonable presentation to occur on a device with fewer colour map entries than the sender had assumed.

If no operand follows a SET COLOUR opcode, the transparent colour is set. If transparent colour is used, 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 analogue 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 colour shows as black.

**D.5.3.2.5.2** The default contents of the colour map are defined according to the algorithm described below:

N = number of bits of the colour map address

$2^N$  = size of the colour map

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

$M \geq 3(N - 1)$  as specified previously

The first half of the default colour map is used to store a complete, uniformly spaced grey scale. This comprises the ordered set of colours where  $G = R = B$ .

(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 colour 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 D.24 and is defined with the three primary colours (green, red, and blue) lying equidistant around the circle with blue at 0 degrees, red at 120 degrees, and green at 240 degrees. 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 GRB values for the default hues, which -lie equally spaced around the hue circle starting at 0 degrees and proceeding counter-clockwise, is as follows:

Let

h the 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 GRB system that must be combined to give the hue h will be:

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

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

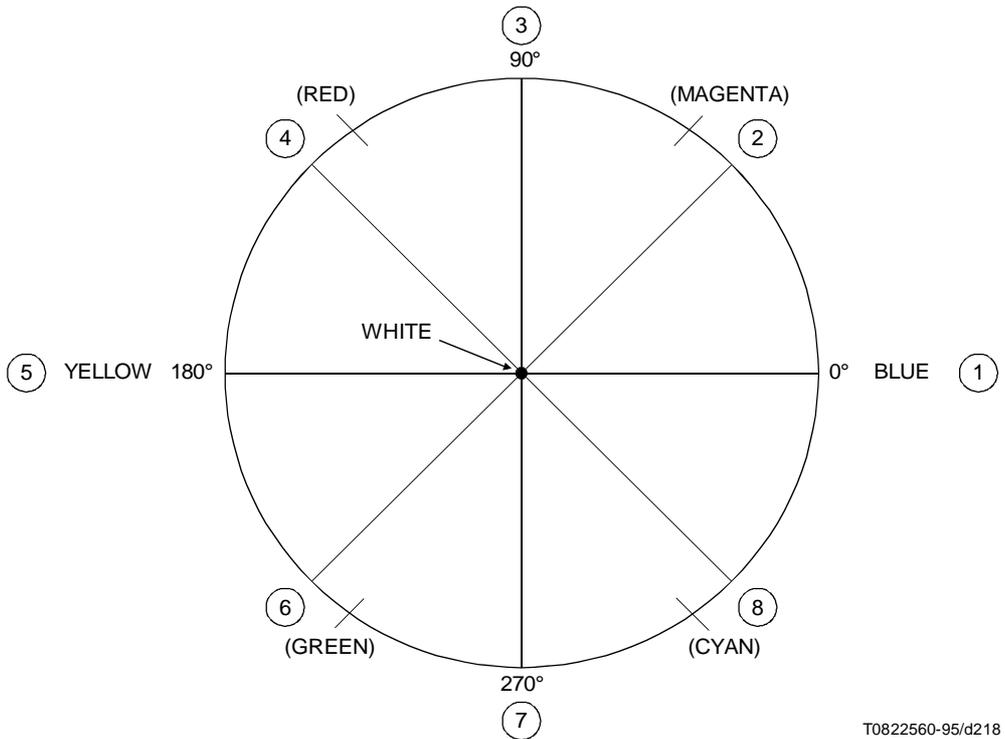
3)  $P_3 = 0$  (i.e. all bits set to 0)

The value of  $P_2$  is then normalized by multiplying it by the maximum colour value which can be stored for that primary. For example, if three bits are available to store the primary, the result given by the above equation for  $P_2$  is multiplied by  $7/8$  (the maximum binary fraction expressible in three bits) and then rounded to three places.

As an example, the default colour map for  $N = 4$  and  $M = 9$  shall be as shown in Figure D.25.

### **D.5.3.2.6 SELECT COLOUR**

**D.5.3.2.6.1** The SELECT COLOUR opcode is used to establish the colour mode as well as select the drawing colour for modes 1 and 2, and the background colour for mode 2 (see Figure D.26).



T0822560-95/d218

Figure D.24/T.101 – Selection of default colours

Colour map address	Colour values		
	G	R	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	0 0 0	1 0 1	1 1 1
1 0 1 0	0 0 0	1 1 1	1 0 0
1 0 1 1	0 1 0	1 1 1	0 0 0
1 1 0 0	1 1 1	1 1 1	0 0 0
1 1 0 1	1 1 1	0 1 0	0 0 0
1 1 1 0	1 1 1	0 0 0	1 0 0
1 1 1 1	1 0 1	0 0 0	1 1 1

Nominal black

↑

Grey scale

↓

Nominal white

↑

Hues

↓

T0822570-95/d219

Figure D.25/T.101 – Default colour map for N = 4, M = 9

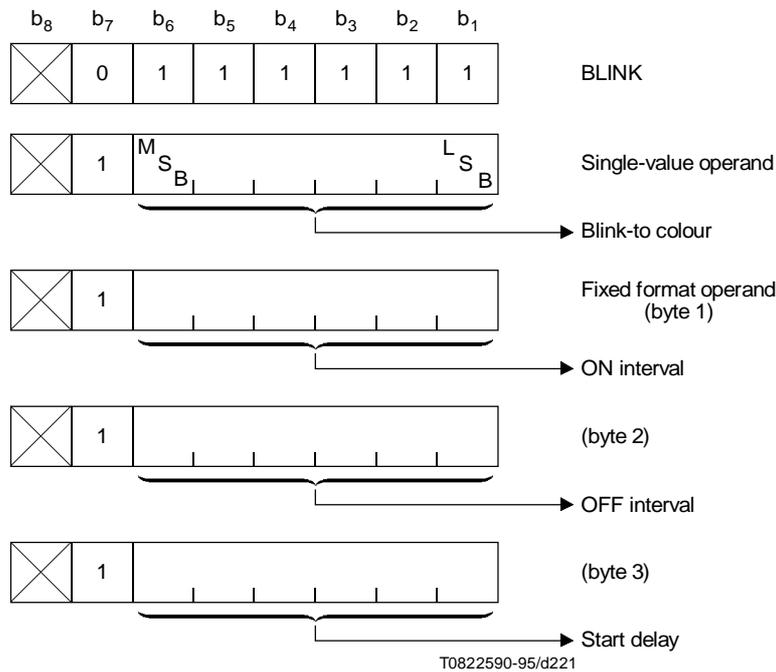


**D.5.3.2.7 BLINK**

**D.5.3.2.7.1** The BLINK command is used to cause a colour map entry to periodically alternate between two colours.

**D.5.3.2.7.2** The mechanism for performing this alternation is called a blink process. This process periodically overwrites the contents of the current in-use drawing colour (the “blink-from” colour) and substitutes the current contents of another entry in the colour map, which is called the “blink-to” colour. The blink-to colour is activated for a period of time known as the ON interval. The blink-from colour is activated for a period of time known as the OFF interval. The ON and OFF intervals alternate with each other, starting with the ON interval. A start 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 most recently defined active blink process. A start delay specification when there are no active blink processes has no effect. 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 colour map that resulted from the previously executed blink process.

**D.5.3.2.7.3** The first single-value operand following the BLINK opcode is the blink-to colour specification, specified as a colour map address (see D.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  $b_6$  through  $b_1$  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 start delay, also in units of 1/10 of a second. If this byte is omitted, a start 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 colour pair (see Figure D.27).



**Figure D.27/T.101 – BLINK**

**D.5.3.2.7.4** Defining a blink process on a pair of blink-to and blink-from colours automatically terminates any previously defined blink process operating on the same pair of colours. If no operands follow the BLINK opcode, then all blink processes utilizing the current drawing colour as the blink-from colour will be terminated. The original blink-from colour shall be restored (unless it has been changed explicitly by a SET COLOUR command) when all blink processes using that blink-from colour are terminated.

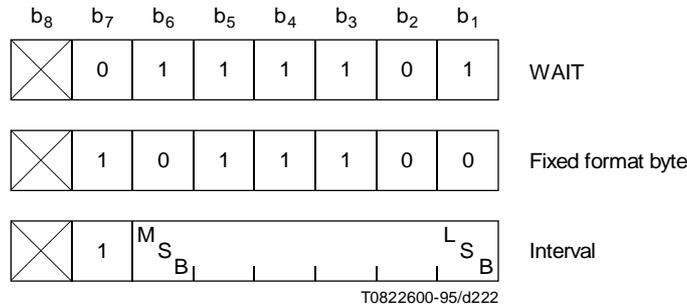
**D.5.3.2.7.5** If additional data follow a completely specified blink process, then the BLINK command is implicitly repeated, with the address of the blink-from colour being automatically incremented (as in D.5.3.2.5.1) prior to the execution of the new opcode. The drawing colour is not affected by this incrementing.

**D.5.3.2.8 WAIT**

**D.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.

**D.5.3.2.8.2** The first byte of operand data following the WAIT opcode shall follow the format given in Figure D.28. If any other bit combination follows the WAIT opcode, the entire command is reserved for future standardization and shall be executed as a null operation. The next byte of operand data gives the time delay in units of 1/10 of a second (64 binary coded values). Only bits b<sub>1</sub> through b<sub>6</sub> 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 between zero and 1/10 of a second (inclusive) that is implementation-dependent.



**Figure D.28/T.101 – WAIT**

**D.5.3.2.9 RESET**

**D.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 colour, home the cursor, and clear the DRCS set, texture attributes, macros, and unprotected fields (described in D.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 (b<sub>1</sub>) to high order bit (b<sub>6</sub>), followed by byte 2, low order bit (b<sub>1</sub>) to high order bit (b<sub>6</sub>). The RESET opcode and its operand are shown below (see Figure D.29).

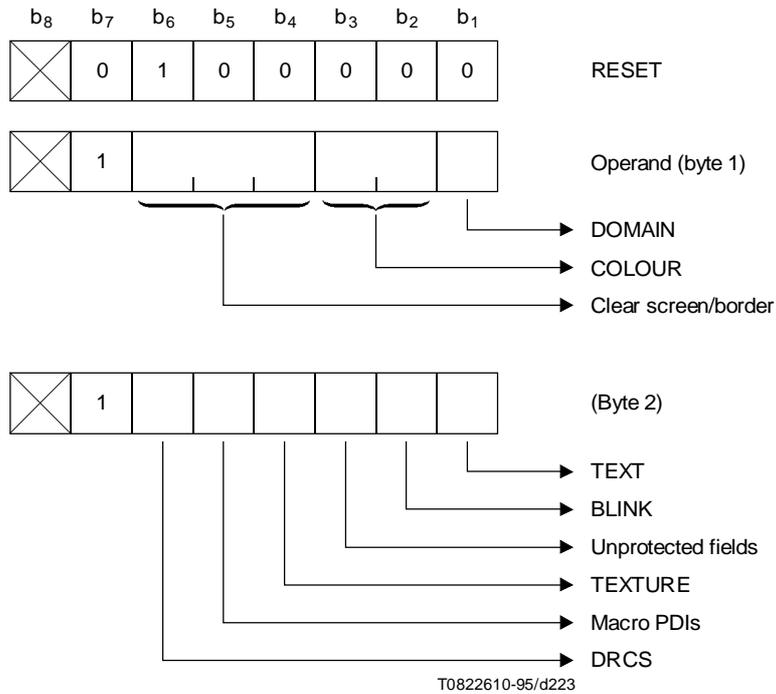
**D.5.3.2.9.2 Operand byte 1 of RESET**

If bit b<sub>1</sub> of byte 1 equals 1, the DOMAIN parameters are reset to their default values. If b<sub>1</sub> is 0, the DOMAIN parameters are not changed.

Bits b<sub>3</sub> and b<sub>2</sub> of byte 1 modify the colour mode and/or current drawing colour as shown in Table D.14.

Bits b<sub>6</sub>, b<sub>5</sub>, and b<sub>4</sub> of byte 1 clear the display area and/or border area to the colours shown in Table D.15.

The border area surrounds the display area and may only be set to one colour at a time.



**Figure D.29/T.101 – RESET**

**Table D.14/T.101 – Color mode reset**

b <sub>3</sub>	b <sub>2</sub>	Color mode
0	0	No action.
0	1	Select colour mode 0, set colour map to default colours and set the in-use drawing colour to white.
1	0	Select colour mode 1 and set colour map to default colours. If this is executed while in colour mode 0, then it has the same effects as “11”.
1	1	Select colour mode 1, set colour map to default colours and set the in-use drawing colour to white.

**D.5.3.2.9.3 Operand byte 2 of RESET**

If bit b<sub>1</sub> of byte 2 equals 1, the cursor is sent to its home position (top left character position in the display area) and all text parameters (from the TEXT opcode, from the C1 set and the active field) are reset to their default values. If b<sub>1</sub> is 0, the text parameters and the cursor position are not changed.

If bit b<sub>2</sub> of byte 2 equals 1, all blink processes are terminated. If b<sub>2</sub> is 0, then blink processes are not changed.

If bit b<sub>3</sub> of byte 2 equals 1, all unprotected fields are changed to protected status but the displayed contents are unaffected. However, the field definitions (except that of the active field) are lost, as well as any data structures maintained for user editing and transmission. If b<sub>3</sub> is 0, unprotected fields are not changed.

If bit b<sub>4</sub> of byte 2 equals 1, all texture attributes are set to their default values. The four programmable texture masks are not cleared. If b<sub>4</sub> is 0, current texture attributes are not changed.

If bit b<sub>5</sub> of byte 2 equals 1, all macros are cleared. This includes transmit-macros. If b<sub>5</sub> is 0, macros are not changed.

**Table D.15/T.101 – Screen and Border Reset**

$b_6$	$b_5$	$b_4$	Screen/Border Colours
0	0	0	No action.
0	0	1	Display area to nominal black.
0	1	0	Display area to current drawing colour.
0	1	1	Border area to nominal black.
1	0	0	Border area to current drawing colour.
1	0	1	Display area and border area to current drawing colour.
1	1	0	Display area to current drawing colour and border area to nominal black.
1	1	1	Display area and border area to nominal black.

If bit  $b_6$  of byte 2 equals 1, all DRCS characters are cleared, that is, all character positions are set to the space character. If  $b_6$  is 0, the DRCS characters are not changed.

If the RESET command is received with no operands, it is interpreted as if it had been sent with bits  $b_6$  to  $b_1$  in both bytes set equal to 0. If only one byte is received, the second operand is then interpreted as if it had been received with bits  $b_6$  to  $b_1$  set equal to 0. If more than two data bytes are received, the additional byte(s) are reserved for future standardization and shall be ignored.

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

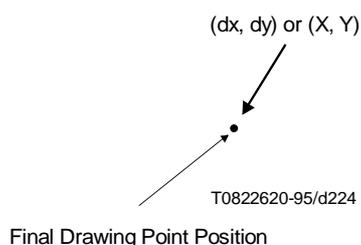
### D.5.3.3 Geometric drawing primitives

Note that the drawings in this subclause are stylized and are not intended to indicate an actual image (see Figure D.15).

#### D.5.3.3.1 POINT

**D.5.3.3.1.1** The POINT command is used to perform two basic geometric drawing operations, that of establishing the coordinate at which to commence drawing and that of drawing a point. A coordinate pair is specified with this command to set the drawing point. Optionally, a point may be drawn (i.e. made visible) at the specified coordinate position. The coordinate is specified either as an absolute (X, Y) position or as a relative (dx, dy) displacement from the current drawing point (see Figure D.30).

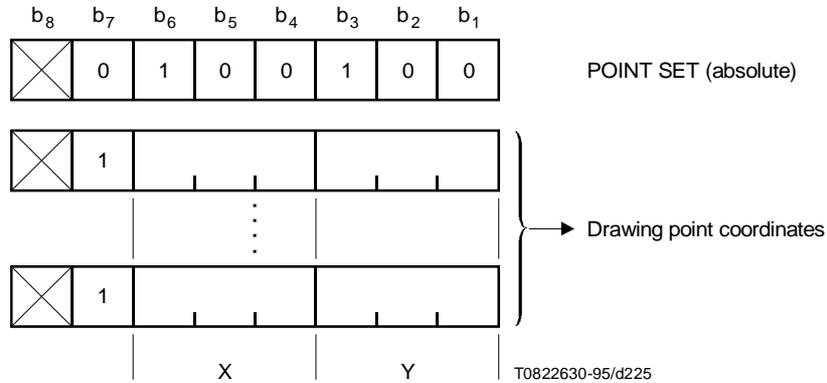
A series of coordinate positions following a POINT opcode may be used to draw a point by point graph. The final drawing point, i.e. the drawing point at the completion of execution of the POINT command, is the last specified point.

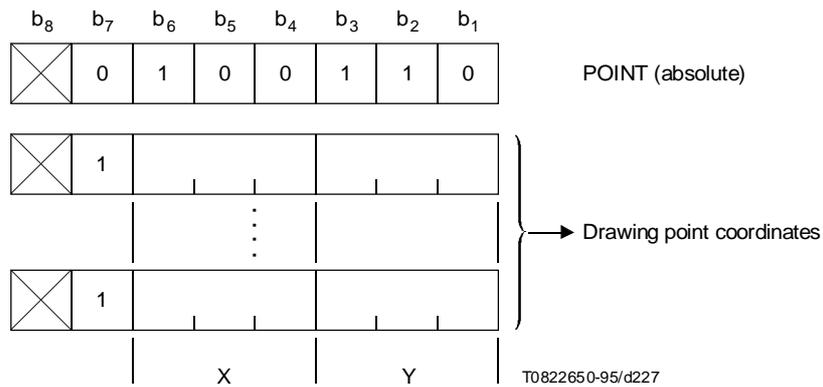


**Figure D.30/T.101 – POINT**

**D.5.3.3.1.2 POINT SET (absolute, invisible)**

This command sets the drawing point to the absolute coordinates specified. A point is not drawn (see Figure D.31).

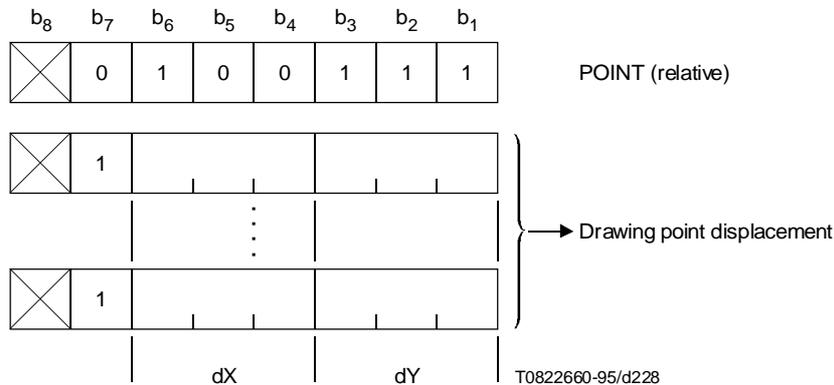




**Figure D.33/T.101 – POINT (absolute, visible)**

#### D.5.3.3.1.5 POINT (relative, visible)

This command sets the drawing point to the coordinates obtained by adding the displacement specified to the coordinates of the current drawing point, and draws a point whose size is determined by the logical pel size, and whose colour is determined by the drawing colour (see Figure D.34).

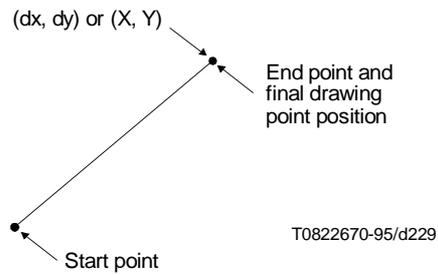


**Figure D.34/T.101 – POINT (relative, visible)**

#### D.5.3.3.2 LINE

**D.5.3.3.2.1** The LINE command is a basic geometric drawing operation. The direction and length of a line are specified by the start and end points. The start point is specified either explicitly within the LINE command or as the current drawing point. The end point is specified either as a relative (dx, dy) displacement from the start point or as an absolute (X, Y) coordinate. At the completion of drawing a line, the drawing point is coincident with the end point. The line is drawn from the start point to the end point in the current colour(s), has a width that is determined by the logical pel size, and a texture determined by the line current texture attribute (see Figure D.35).

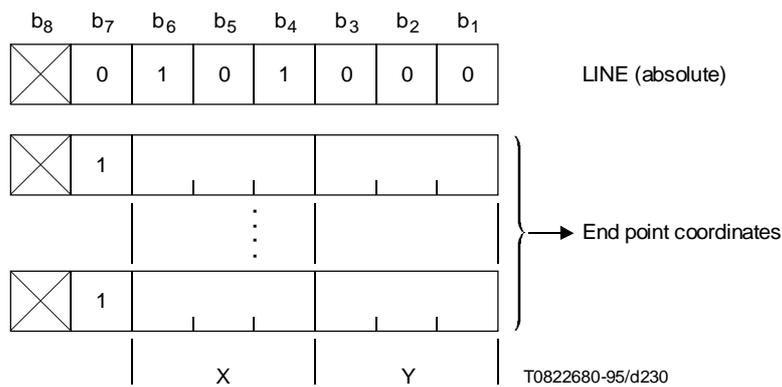
The LINE command may be used to draw a line graph from a table of numbers described as absolute or relative coordinates in the same manner as the POINT opcode.



**Figure D.35/T.101 – LINE**

**D.5.3.3.2.2 LINE (absolute)**

The start point is the current drawing point. The end point is specified in absolute coordinates (see Figure D.36).



**Figure D.36/T.101 – LINE (absolute)**

**D.5.3.3.2.3 LINE (relative)**

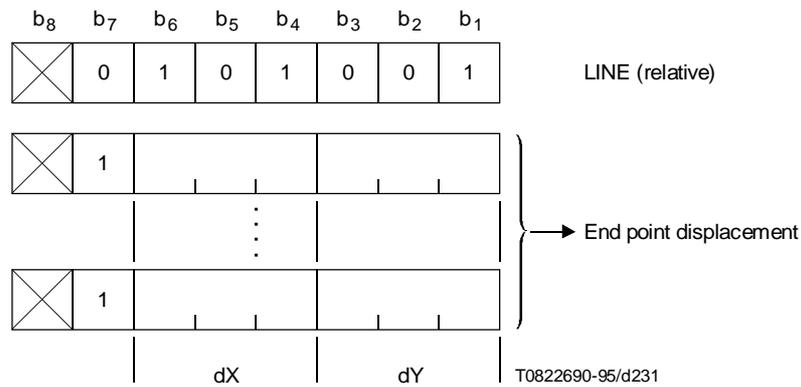
The start point is the initial drawing point. The end point is specified as a relative displacement from the start point. (see Figure D.37).

**D.5.3.3.2.4 SET and LINE (absolute)**

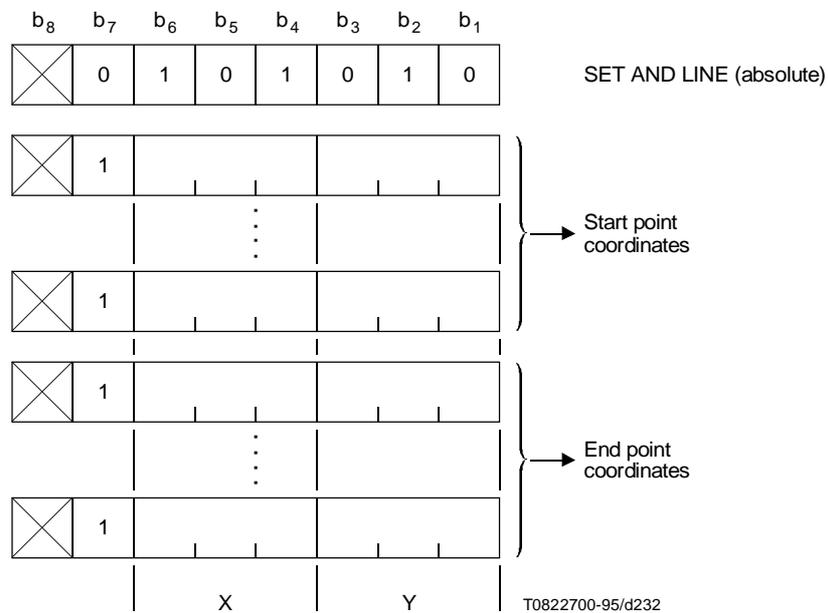
Both the start and end points are specified in absolute coordinates (see Figure D.38). If more than two operands are present, lines are drawn from the first to the second point, then from the third to the fourth point, etc.

**D.5.3.3.2.5 SET and LINE (relative)**

The start point is specified in absolute coordinates, and the end point is specified as a relative displacement from the start point (see Figure D.39).



**Figure D.37/T.101 – LINE (relative)**

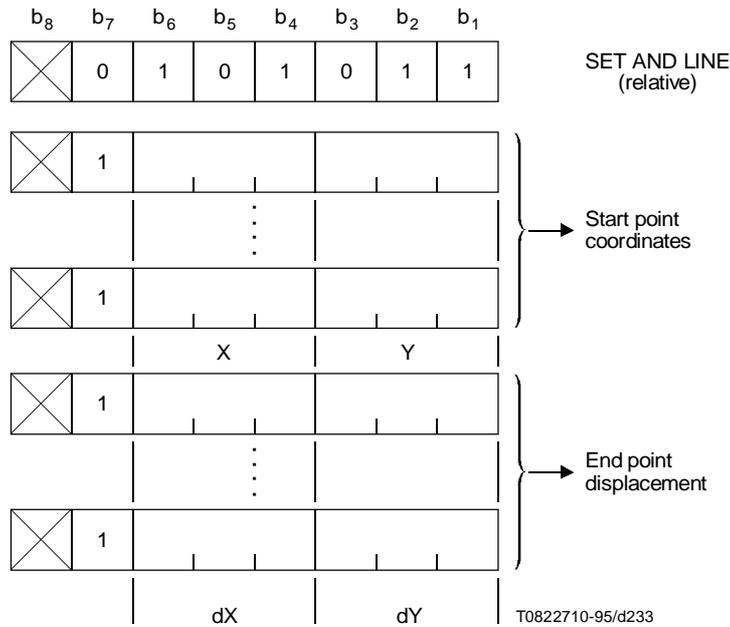


**Figure D.38/T.101 – SET and LINE (Absolute)**

### D.5.3.3.3 ARC

**D.5.3.3.3.1** The ARC geometric drawing operation provides the capability of drawing circles, segments of circles, and curvilinear splines. For circles and segments of circles, an arc is drawn from a start point to an end point through an intermediate point on the arc. Drawing of a circle results when the start and end points are coincident; the intermediate point defines the diameter of the circle, and therefore is the midpoint on the arc between the start and end points. A segment of a circle is drawn when the start and end points are not coincident.

The start point is specified either explicitly within the ARC command or as the current drawing point. The intermediate point is described as a relative displacement from the start point. The end point is specified as a relative displacement from the intermediate point. 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.

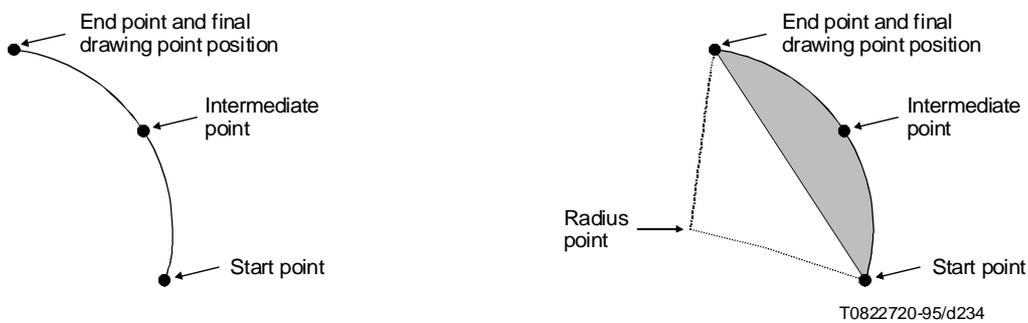


**Figure D.39/T.101 – SET and LINE (relative)**

If the three drawing points are colinear, a line is drawn from the start point to the end point, except for the error condition in which the intermediate point does not lie between the start and end points. If the end point 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 D.5.3.1.1). At the completion of drawing an arc, the drawing point is coincident with the end point.

An arc may be either filled or outlined. Outlined arcs are drawn in the current colour(s), have a width that is determined by the logical pel size, and a line texture that is as specified by the TEXTURE command. The chord that joins the start and end points is not considered part of the outline and, as such, is not drawn.

For filled arcs, the area enclosed by the outline and the chord (including the region of the outline and the chord traced by the logical pel) is filled in the current colour(s) with the texture pattern specified in the TEXTURE command. The stroke width of the chord is affected by the logical pel, but the chord is not considered a part of the arc and, as such, is not highlighted if highlight mode is selected (see D.5.3.2.4.3 and Figure D.40).

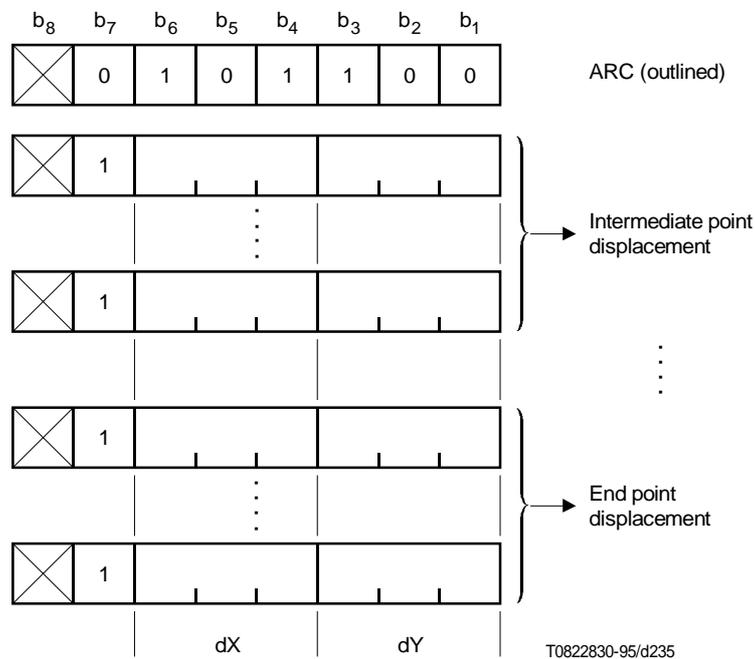


**Figure D.40/T.101 – ARC**

Drawing of a curvilinear spline results when more than three points are specified. The last point specified is the end point. The drawing point at the completion of drawing a spline is the end point. The minimum implementation of the spline shall be a series of lines connecting the start, intermediate, and end points of the spline. The display device may draw a smoother spline, but the shape of this spline and the characteristics of the algorithm used are implementation-dependent. The complete algorithm for a curvilinear spline is reserved for future standardization. All the attributes described above for circles and segments of circles (colinear points, points outside the unit screen, fill, and outline) apply to splines. In the case of a filled spline, the spline and the chord (the line that joins the start and end points) must enclose a single area, i.e. no portion of the spline outline or chord may cross any other portion of the spline or chord. The maximum number of points permitted to describe a spline is implementation dependent, and shall be at least 256 points.

**D.5.3.3.3.2 ARC (outlined)**

The start point is the current drawing point, the intermediate point is the first block of coordinate data, specified as a relative displacement from the start point, and the end point is the second block of coordinate data, specified as a relative displacement from the intermediate point (see Figure D.41). The arc is not filled.



**Figure D.41/T.101 – ARC (outlined)**

**D.5.3.3.3.3 ARC (filled)**

The start point is the current drawing point, the intermediate point is the first block of coordinate data, specified as a relative displacement from the start point, and the end point is the second block of coordinate data, specified as a relative displacement from the intermediate point. The start and end points are joined by a chord and the resulting figure is filled in the current colour(s) with the current texture pattern (see Figure D.42).

**D.5.3.3.3.4 SET and ARC (outlined)**

The start point is the first block of coordinate data, specified in absolute coordinates. The intermediate point is the second block of coordinate data, specified as a relative displacement from the start point, and the end point is the third block of coordinate data, specified as a relative displacement from the intermediate point (see Figure D.43). The arc is not filled.

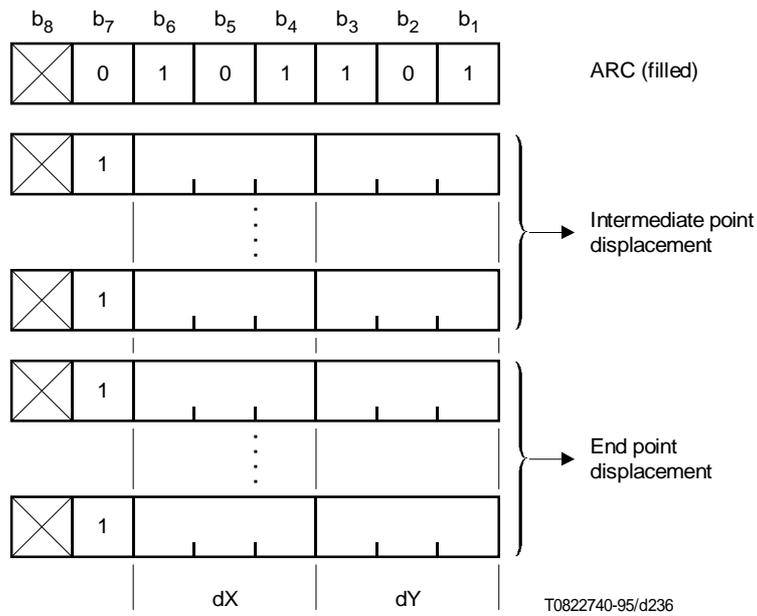


Figure D.42/T.101 – ARC (filled)

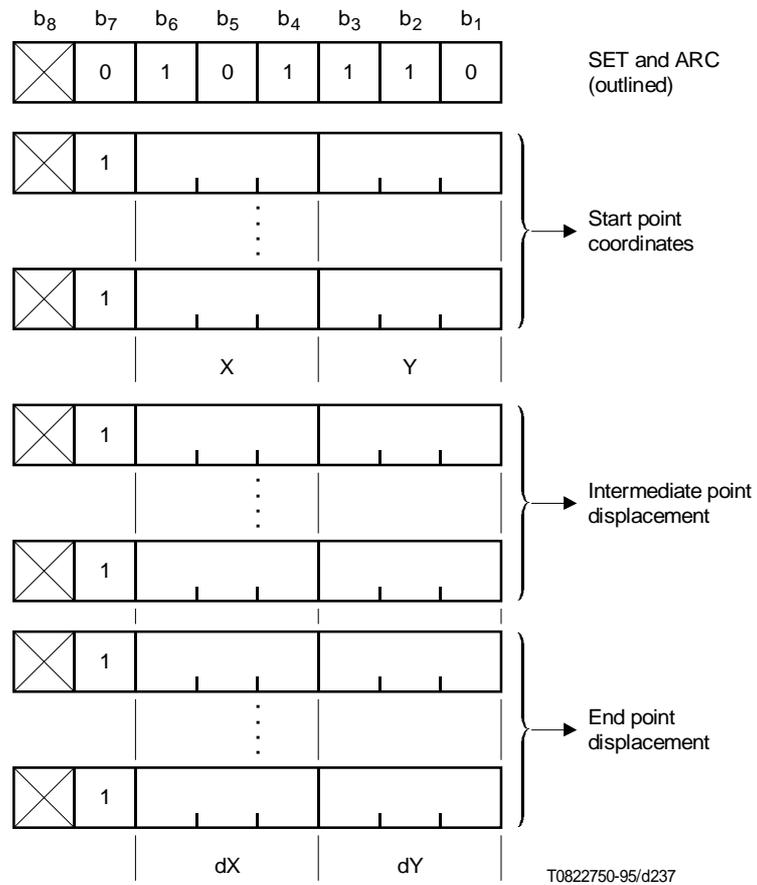
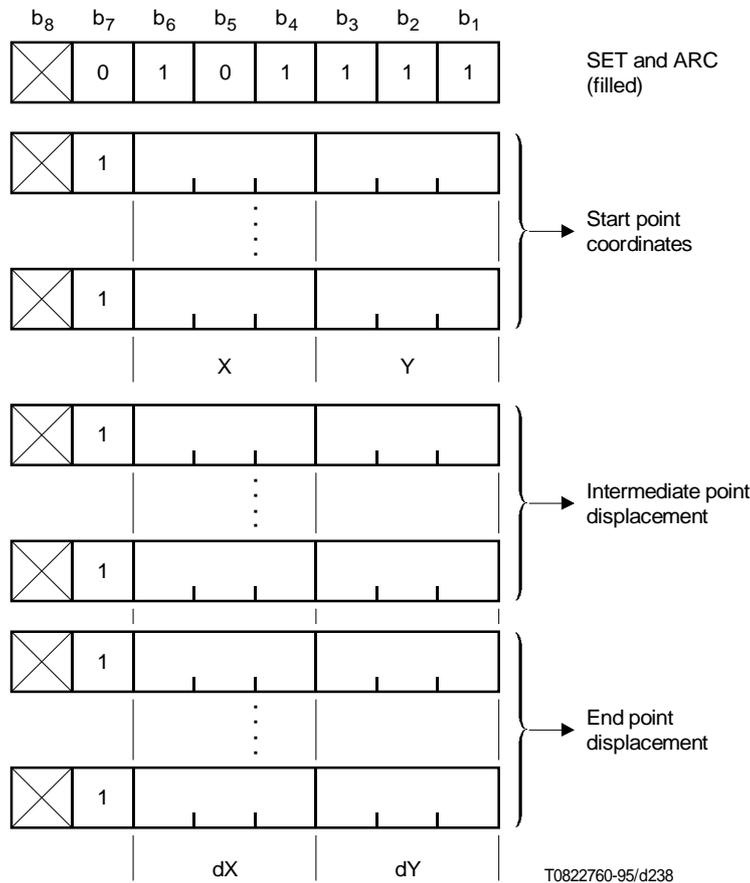


Figure D.43/T.101 – SET and ARC (outlined)

**D.5.3.3.5 SET and ARC (filled)**

The start point is the first block of coordinate data, specified in absolute coordinates. The intermediate point is the second block of coordinate data, specified as a relative displacement from the start point, and the end point is the third block of coordinate data, specified as a relative displacement from the intermediate point (see Figure D.44). The start and end points are joined by a chord and the resulting figure is filled in the current colour(s) with the current texture pattern.



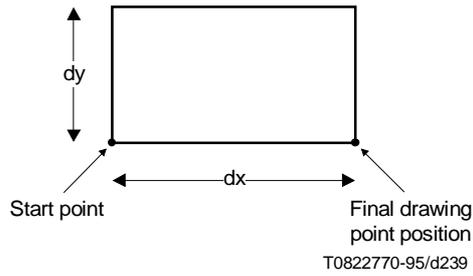
**Figure D.44/T.101 – SET and ARC (filled)**

**D.5.3.3.4 RECTANGLE**

**D.5.3.3.4.1** The RECTANGLE command provides the capability of drawing a rectangular area of width dx and height dy. The start point is specified either explicitly within the RECTANGLE command or as the current drawing point. At the completion of drawing a rectangle, the drawing point is the start point 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 current colour(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 enclosed by the outline (including the region of the outline traced by the logical pel) is filled in the current colour(s) with the texture pattern specified in the TEXTURE command, and the outline is highlighted if the highlight mode is selected (see D.5.3.2.4.3).

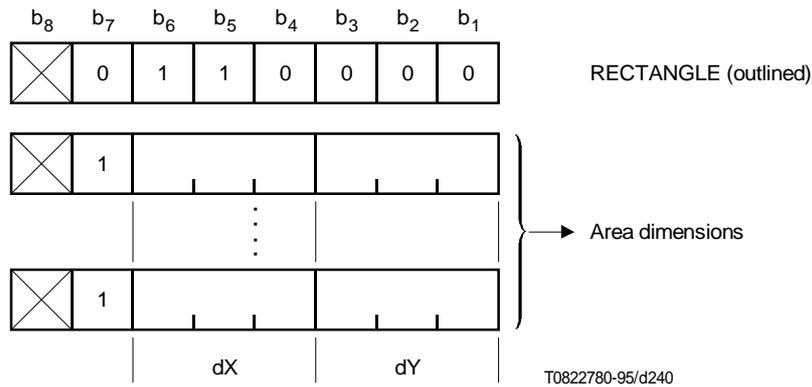
The RECTANGLE command may be used to draw a histogram from a table of numbers representing relative dy and dx displacements in the same manner as the POINT and LINE commands may be used to plot graphs (see Figure D.45).



**Figure D.45/T.101 – RECTANGLE**

**D.5.3.3.4.2 RECTANGLE (outlined)**

The start point is the current drawing point and the width and height (dx, dy) are given as the first block of coordinate data (see Figure D.46). The rectangle is not filled.



**Figure D.46/T.101 – RECTANGLE (outlined)**

**D.5.3.3.4.3 RECTANGLE (filled)**

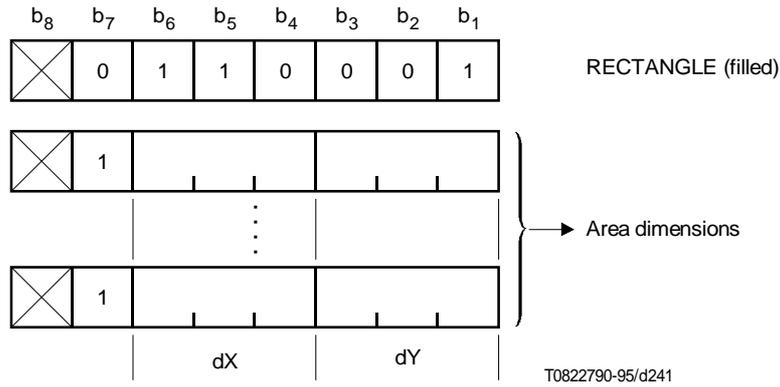
The start point is the current drawing point and the width and height (dx, dy) are given as the first block of coordinate data (see Figure D.47). The rectangle is filled in the current colour(s) with the current texture pattern.

**D.5.3.3.4.4 SET and RECTANGLE (outlined)**

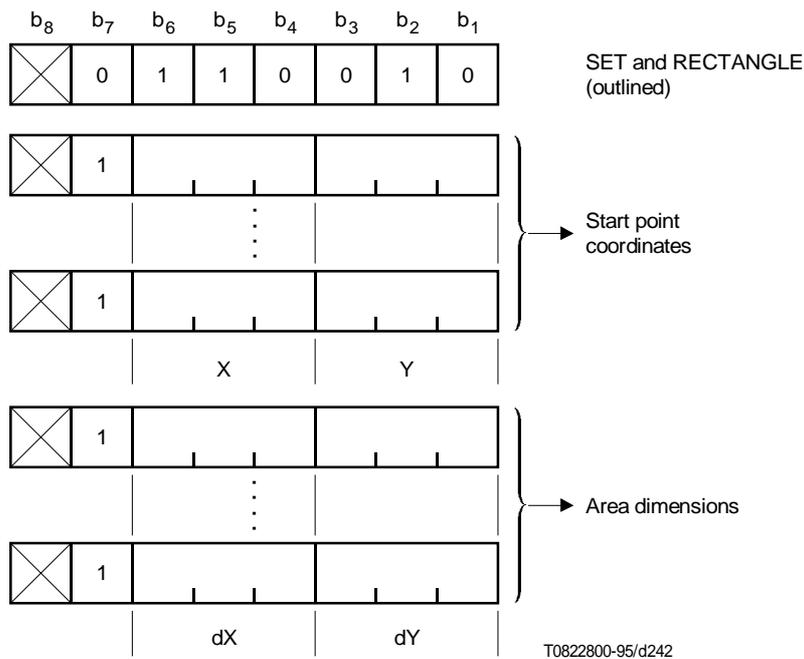
The start point is specified in absolute coordinates as the first block of coordinate data, and the width and height (dx, dy) are given as the second block of coordinate data (see Figure D.48). The rectangle is not filled.

**D.5.3.3.4.5 SET and RECTANGLE (filled)**

The start point 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 (see Figure D.49). The rectangle is filled in the current colour(s) with the current texture pattern.



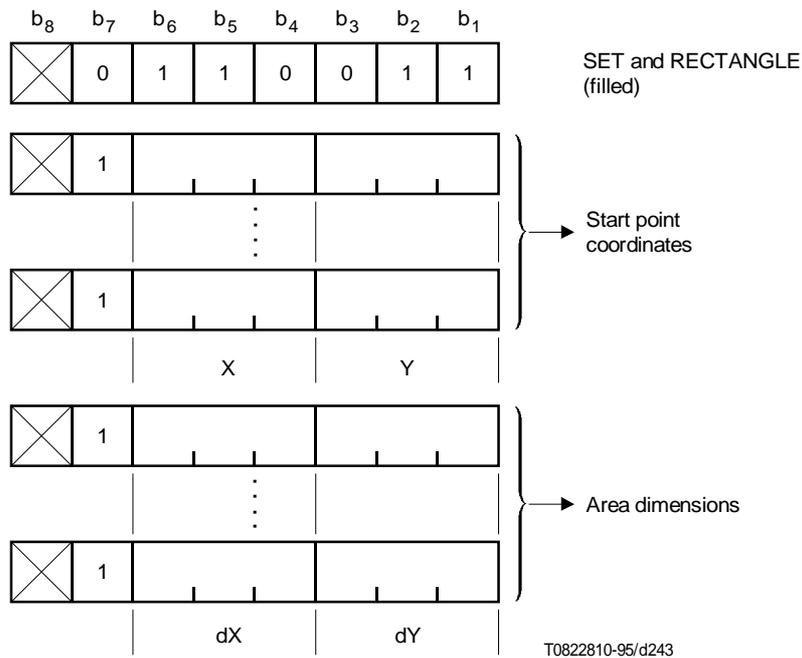
**Figure D.47/T.101 – RECTANGLE (filled)**



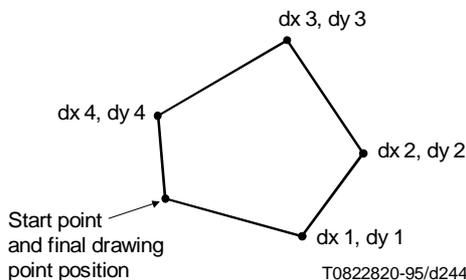
**Figure D.48/T.101 – SET and RECTANGLE (outlined)**

### D.5.3.3.5 POLYGON

**D.5.3.3.5.1** The POLYGON command 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. The start point is specified either explicitly within the POLYGON command or as the current drawing point. 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 start point and the last vertex specified so that at the completion of drawing a polygon, the drawing point is coincident with the start point (see Figure D.50).



**Figure D.49/T.101 – SET and RECTANGLE (filled)**



**Figure D.50/T.101 – POLYGON**

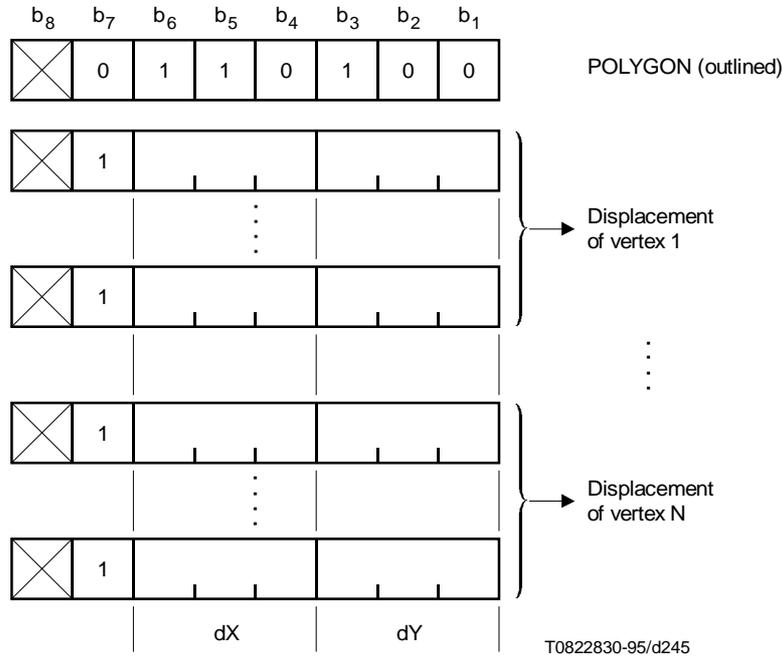
A polygon may be either filled or outlined. Outlined polygons are drawn in the current colour(s) and have a line width that is determined by the logical pel size, and a line texture that is as specified by TEXTURE command. For filled polygons, the area enclosed by the outline (including the region of the outline traced by the logical pel) is filled in the current colour(s) with the texture pattern specified in the TEXTURE command, and the outline is highlighted if the highlight mode is selected (see D.5.3.2.4.3).

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 implementation dependent and shall be at least 256 vertices.

**D.5.3.3.5.2 POLYGON (outlined)**

The start point is the current drawing point and subsequent vertex coordinates are specified as relative displacements from the previous vertex coordinate (see Figure D.51). The polygon is not filled.



**Figure D.51/T.101 – POLYGON (outlined)**

**D.5.3.3.5.3 POLYGON (filled)**

The start point is the current drawing point and subsequent vertex coordinates are specified as relative displacements from the previous vertex coordinate (see Figure D.52). The polygon is filled in the current colour(s) with the current texture pattern.

**D.5.3.3.5.4 SET and POLYGON (outlined)**

The start point 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 (see Figure D.53). The polygon is not filled.

**D.5.3.3.5.5 SET and POLYGON (filled)**

The start point 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 (see Figure D.54). The polygon is filled in the current colour(s) with the current texture pattern.

**D.5.3.3.6 INCREMENTAL**

**D.5.3.3.6.1** The INCREMENTAL command allows 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).

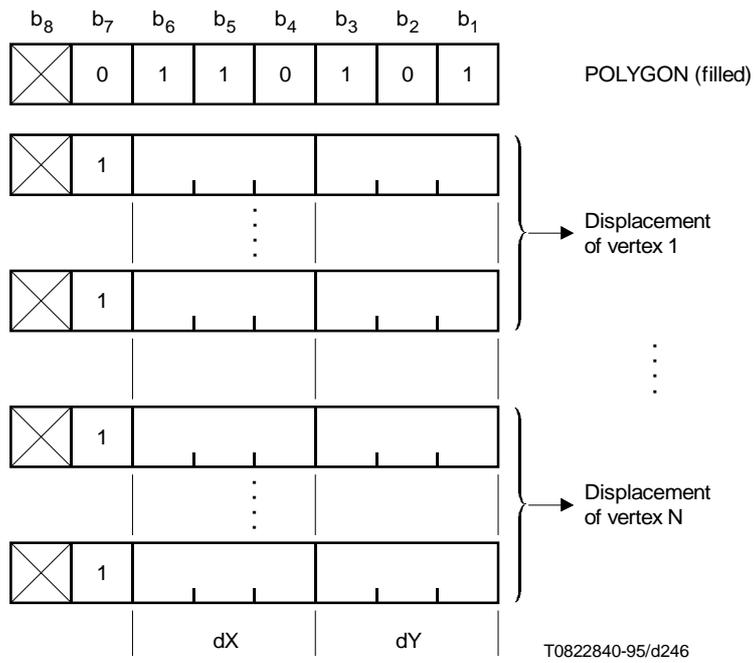


Figure D.52/T.101 – POLYGON (filled)

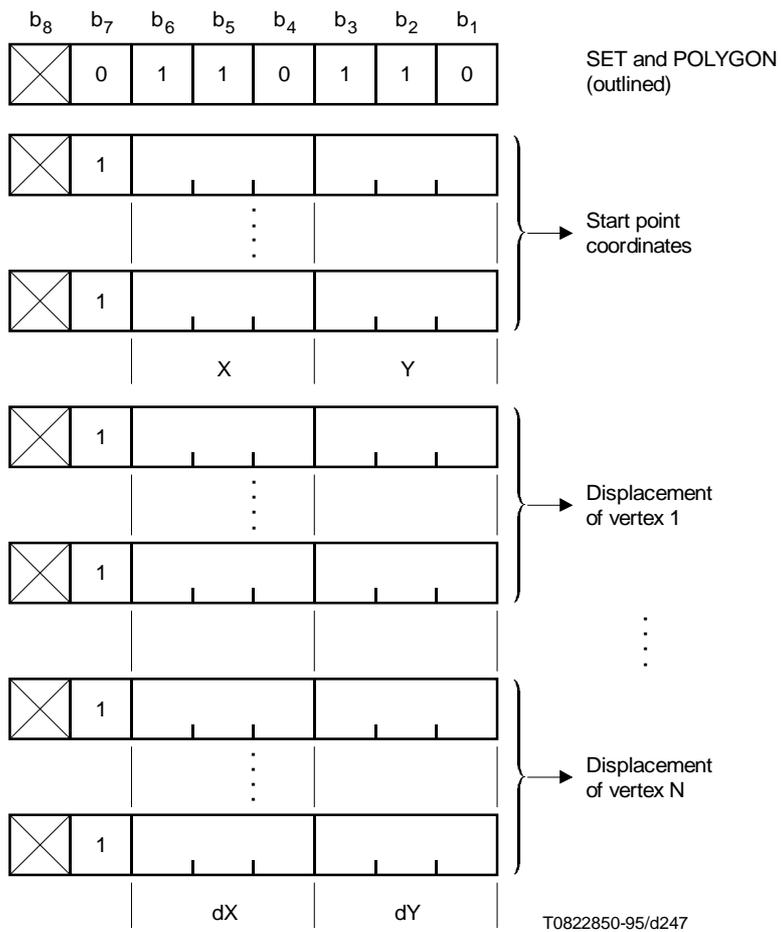
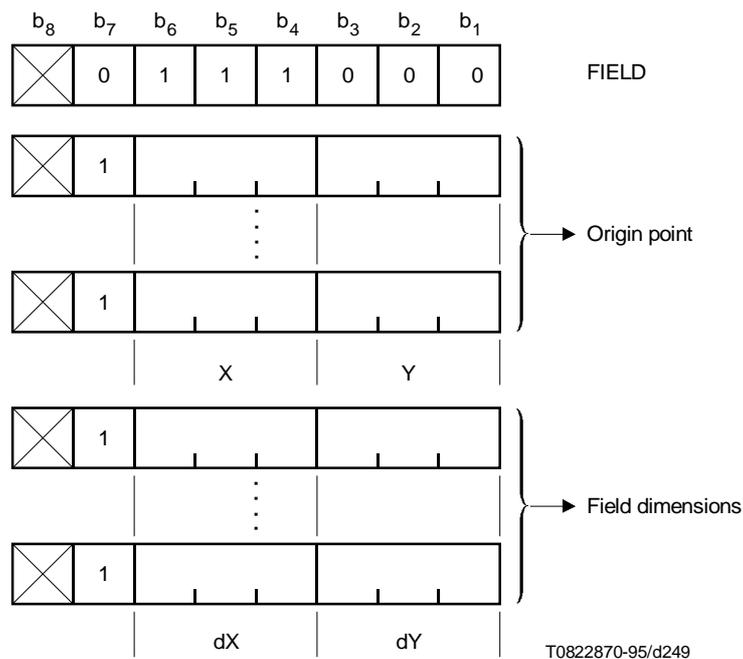


Figure D.53/T.101 – SET and POLYGON (outlined)



- 2) If no portion of the logical pel exceeds the active field, then the first colour 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 colour 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 colour is then obtained and the process is repeated.
  
- 3) If any portion of the logical pel exceeds the active 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 colour specification. Interpretation resumes at the first bit (i.e. b<sub>6</sub>) 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, then the Y value is left constant and the entire display image lying within the area of the screen defined by the active field 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 then set to the origin of the active field.



**Figure D.55/T.101 – FIELD**

The INCREMENTAL POINT opcode and its operands are shown in Figure D.56. An example of an INCREMENTAL POINT picture is shown in Figure D.57.

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 colour specification. The range of values of the packing counter shall be 1 to 48, inclusive. Values of 0 or greater than 48 are reserved for future standardization and the entire INCREMENTAL POINT command containing packing counters with such values shall be executed as a null operation. The second operand is a string operand of indeterminate length. It contains the colour specifications, stored sequentially

without regard to byte boundaries, high order bit ( $b_8$ ) to low order bit ( $b_1$ ) within the numeric data fields. In colour mode 0, these colour specifications are interpreted as actual colour values; that is, a number of bits equal to the packing count is used to define the colour applied to each drawing operation. As in the multi-value colour specification, the bits are organized into three-tuples and the order of interpretation of the bits is G, R, B. Note that a single colour specification may contain multiple three-tuples depending on the packing count, in which case the first three-tuple contains the MSB's and the last contains the LSB's of the three primaries. For example, if the packing count were 6, the colour 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 colour 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 ...

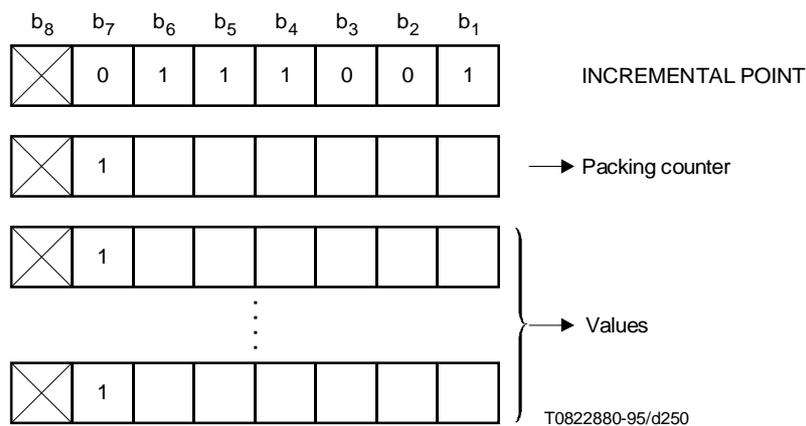


Figure D.56/T.101 – INCREMENTAL POINT



Figure D.57/T.101 – Example INCREMENTAL POINT picture

In colour modes 1 and 2, these colour specifications are ordinal numbers, i.e. addresses in the colour map in which the actual colour value was previously, or will later be, loaded. Again, a number of bits equal to the packing count is used to define the colour applied to each drawing operation. These specifications are concatenated in the string operand without regard to byte boundaries.

It is required, depending on the relationship between the logical pel dimensions and the physical pixel dimensions on an individual display device, to perform a preexecution rescaling of both the logical pel dimensions and the active field dimensions to avoid certain types of distortions (i.e. skew) that will result from a mismatch. When the drawing point is in such an active field, the relative position of the drawing point inside the scaled field should remain the same as before scaling. The goal of this type of rescaling is to make the logical pel dimensions become either integer multiples or integer fractions of the corresponding physical pixel dimensions. (The active field dimensions would have to be scaled equivalently in order to prevent skew in the image.) The following conditions are required:

- 1) the logical pel dimensions and active field dimensions are restored to their prescaled values after the INCREMENTAL POINT command completes execution;
- 2) the resultant image is guaranteed to lie within the original active field; and
- 3) the implementation ensures that skew 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 if any portion of the logical pel indicated by the initial drawing point lies outside the active field, the entire command is considered to be in error and 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 +0.00000001, i.e. 1/256.)

#### D.5.3.3.6.4 INCREMENTAL LINE

The INCREMENTAL LINE command provides the capability of compactly describing an image consisting of a series of short line segments drawn in the current colour(s) and line texture (see Figures D.58 and D.59).

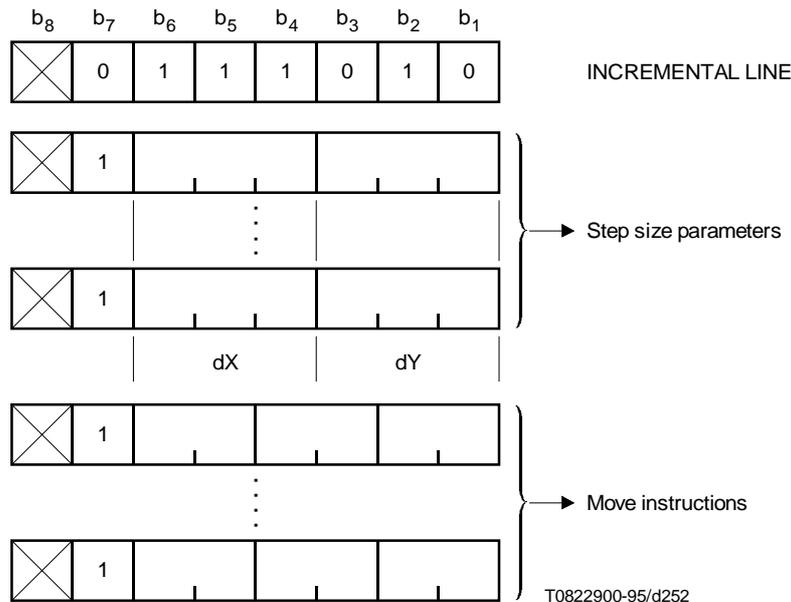
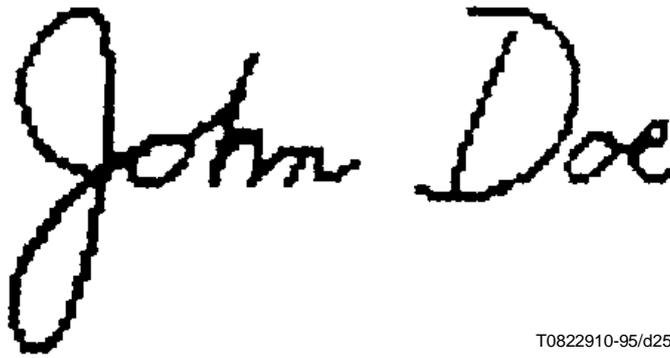


Figure D.58/T.101 – INCREMENTAL LINE



**Figure D.59/T.101 – 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  $b_6$  to  $b_1$ . The interpretation of these two-bit nibbles is as shown in Table D.16.

**Table D.16/T.101 – INCREMENTAL LINE move values**

Nibble value	Interpretation
00	Interpret the following nibble as a modify parameter instruction (see Table D.17).
01	Advance the drawing point a distance dx in the x direction and optionally draw a line.
10	Advance the drawing point a distance dy in the y direction and optionally draw a line.
11	Advance the drawing point a distance dx in the x direction and du 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 colour 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 (see Table D.16), the next nibble is interpreted as a modify parameter instruction as shown in Table D.17. The nibble following that is interpreted as a step operation (see Table D.16). The draw flag is initially on whenever the INCREMENTAL LINE opcode is encountered. When the string operand is terminated, the drawing point is left at the final drawing point.

**Table D.17/T.101 – INCREMENTAL LINE modify parameter instructions**

Nibble value	Modify parameter instruction
00	Change the state of the draw flag (ON to OFF or OFF to ON)
01	Change sign of dx
10	Change sign of dy
11	Change sign of dx and dy

### D.5.3.3.6.5 INCREMENTAL POLYGON (filled)

The INCREMENTAL POLYGON command provides the capability of compactly describing a filled polygon drawn with a series of short line segments. (See Figures D.60 and D.61). The area enclosed by the outline (including the region of the outline traced by the logical pel) is filled in the current colour(s) with the texture pattern specified in the TEXTURE command, and the outline is highlighted if the highlight mode is selected (see D.5.3.2.4.3).

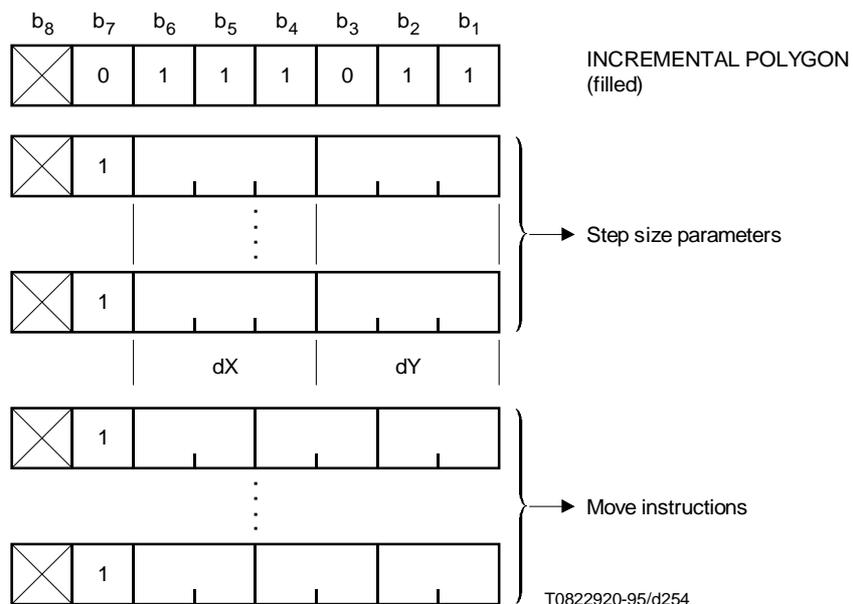


Figure D.60/T.101 – INCREMENTAL POLYGON (filled)



Figure D.61/T.101 – Example INCREMENTAL POLYGON picture

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

- 1) The draw flag is always on.
- 2) Should a “modify parameter” instruction (see Table D.17) 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.
- 3) The final drawing point is implicitly taken as the initial drawing point.
- 4) The resulting figure is filled in the current colour and texture pattern and according to the highlight attribute.

Note that INCREMENTAL POLYGON shall enclose a single area (similar to POLYGON, see D.5.3.3.5).

#### D.5.4 Mosaic set

Figure D.63 shows the character assignments for the mosaic set. This comprises sixty-five  $2 \times 3$  block mosaic characters including a second copy of the solid mosaic in position 5/15. The remaining character positions are reserved for future standardization and shall be displayed as SPACE. Mosaic characters can be displayed in two modes, contiguous or separated, depending on the underline mode described in D.6.2.7.15. In contiguous mode, the six mosaic elements that compose each character shall 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 absolute value of the width (dx) of the logical pel size, and are reduced in the vertical dimension by the absolute value of the height (dy) of the logical pel size. (The logical pel is described in D.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. If either dimension of the logical pel is equal to or exceeds the corresponding dimension of the mosaic element, then the illuminated area is reduced to zero. 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 D.62)

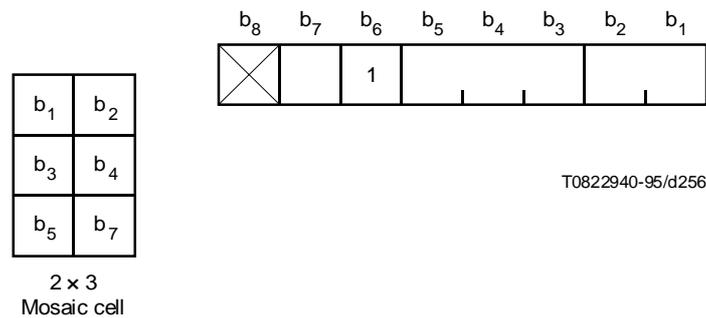


Figure D.62/T.101 – Mosaic sub-element encoding

The code combination 5/15 (1011111) also implies that all sub-elements are on. See Figure D.63 for a code table representation of the mosaic scheme.

The sequence used to designate the mosaic set is ESC I 7/13, where I is 2/9 or 2/13 to indicate G1, 2/10 or 2/14 to indicate G2, or 2/11 or 2/15 to indicate G3 (see D.4.3).

					10	11	12	13	14	15	
					b <sub>7</sub>	0	0	1	1	1	1
					b <sub>6</sub>	1	1	0	0	1	1
					b <sub>5</sub>	0	1	0	1	0	1
					Column	2	3	4	5	6	7
b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	Row							
0	0	0	0	0							
0	0	0	1	1							
0	0	1	0	2							
0	0	1	1	3							
0	1	0	0	4							
0	1	0	1	5							
0	1	1	0	6							
0	1	1	1	7							
1	0	0	0	8							
1	0	0	1	9							
1	0	1	0	10							
1	0	1	1	11							
1	1	0	0	12							
1	1	0	1	13							
1	1	1	0	14							
1	1	1	1	15							

T0822950-95/d257

NOTE – The rectangles surrounding the characters are for illustrative purposes only and are not part of the graphic symbols.

Figure D.63/T.101 – Mosaic set

### D.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 layer 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 D.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 layer assume responsibility for infinite loops.

Any macro (whether defined by DEF MACRO, DEFP MACRO, or DEFT MACRO) (see D.6.2.2) may be linked to a user input mechanism, such as a function key, to allow its execution or transmission to be activated by the user. The number of macro names that can be linked is implementation-dependent.

The sequence used to designate the macro set is ESC I 7/10, where I is 2/9 or 2/13 to indicate G1, 2/10 or 2/14 to indicate G2, or 2/11 or 2/15 to indicate G3 (see D.4.3).

### **D.5.6 Dynamically Redefinable Character Set (DRCS)**

Unlike the other character sets, whose pattern definitions are permanently stored in the receiving device and cannot be altered by the information provider, 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 a similar manner as the primary, supplementary, and mosaic sets. At display time, they are subject to the same attributes as alphanumeric text. The manner in which the patterns are actually downloaded is described in D.6.2.3. If a DRCS character has not been defined by the DEF DRCS command, it shall be displayed as if it were SPACE.

The sequence used to designate the DRCS set is ESC I 7/11, where I is 2/9 or 2/13 to indicate G1, 2/10 or 2/14 to indicate G2, or 2/11 or 2/15 to indicate G3 (see D.4.3).

## **D.6 Coding of C-sets**

### **D.6.1 C0 control set**

**D.6.1.1** This clause describes the C0 control set (see Figure D.64) that occupies columns 0 and 1 of the 7- and 8-bit in-use tables. The functions are as follows.

#### **D.6.1.2 Format effector characters**

##### **D.6.1.2.1 ACTIVE POSITION BACKWARD (APB)**

This character (0/8) (backspace) is used to move the cursor a distance equal to the intercharacter spacing lying parallel to the character path in the direction opposite to the character path (i.e. 180 degrees from the direction of the character path). If such a movement would cause any part of the full corresponding character field to be outside of the display area (or outside of the active field, if the character field was entirely within the active field immediately before the movement), then the cursor is instead moved to the opposite edge (along the character path) of the display area (or active field) and an automatic APU is executed.

##### **D.6.1.2.2 ACTIVE POSITION FORWARD (APF)**

This character (0/9) (horizontal tab) is used to move the cursor a distance equal to the intercharacter spacing lying parallel to the character path in the direction of the character path. If such a movement would cause any part of the full corresponding character field to be outside of the display area (or outside of the active field, if the character field was entirely within the active field immediately before the movement), then the cursor is instead moved to the opposite edge (along the character path) of the display area (or active field) and an automatic APD is executed.

##### **D.6.1.2.3 ACTIVE POSITION DOWN (APD)**

This character (0/10) (line feed) is used to move the cursor a distance equal to the interrow space lying perpendicular to the character path in a direction perpendicular to the character path (-90 degrees). If such a movement would cause any part of the full corresponding character field to be outside of the display area (or outside of the active field, if the character field was entirely within the active field immediately before the movement), then special action is taken that is dependent on whether or not scroll mode is in effect (see 6.2.7.13 and D.6.2.7.14).

##### **D.6.1.2.4 ACTIVE POSITION SET (APS)**

This character (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 or 10 through 15 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  $b_7$  through  $b_1$  with  $b_7$  being the MSB, masking out  $b_8$ , and subtracting 32. Similarly, the column address is obtained from the second byte following an APS by taking the binary integer comprising bits  $b_7$  through  $b_1$  with  $b_7$  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. If either of the characters following the APS character is a C0 or C1 control, the APS is ignored and the C0 or C1 control is executed.

Rows and columns are numbered starting with row 0, column 0, in the lower leftmost character position of the display area, and refer to the nominal screen format established by the current character field size (with the default intercharacter and interrow 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.

					b <sub>7</sub>	0	0
					b <sub>6</sub>	0	0
					b <sub>5</sub>	0	1
					Column	0	1
b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	Row			
0	0	0	0	00	NUL	DLE	
0	0	0	1	01	SOH	DC1	
0	0	1	0	02	STX	DC2	
0	0	1	1	03	ETX	DC3	
0	1	0	0	04	EOT	DC4	
0	1	0	1	05	ENQ	NAK	
0	1	1	0	06	ACK	SYN	
0	1	1	1	07	BEL	ETB	
1	0	0	0	08	APB (BS)	CAN	
1	0	0	1	09	APF (HT)	SS2	
1	0	1	0	10	APD (LF)	SDC	
1	0	1	1	11	APU (VT)	ESC	
1	1	0	0	12	CS (FF)	APS	
1	1	0	1	13	APR (CR)	SS3	
1	1	1	0	14	SO	APH	
1	1	1	1	15	SI	NSR	

T0823140-95/d258

**Figure D.64/T.101 – C0 control set**

#### D.6.1.2.5 ACTIVE POSITION UP (APU)

This character (0/11) (vertical tab) is used to move the cursor a distance equal to the interrow space lying perpendicular to the character path in a direction perpendicular to the character path (90 degrees). If such a movement would cause any part of the full corresponding character field to be outside of the display area (or outside of the active field, if the character field was entirely within the active field immediately before the movement), then special action is taken that is dependent on whether or not scroll mode is in effect (see D.6.2.7.13 and D.6.2.7.14).

#### **D.6.1.2.6 CLEAR SCREEN (CS)**

This character (0/12) is used to move the cursor to the upper left character position of the display area, in which the top of the character field coincides with the top boundary of the display area. In colour modes 0 and 1, it clears the display area to nominal black. In colour mode 2, it clears the display area to the background colour.

#### **D.6.1.2.7 ACTIVE POSITION RETURN (APR)**

This character (0/13) (carriage return) is used to move the cursor to the first character position within the display area (or within the active field, should the full character field corresponding to the cursor lie entirely within the active field before the movement) along the character path.

#### **D.6.1.2.8 ACTIVE POSITION HOME (APH)**

This character (1/14) is used to move the cursor to the upper left character position in the display area, in which the top of the character field coincides with the top boundary of the display area.

### **D.6.1.3 Code extension control characters**

#### **D.6.1.3.1 SHIFT-OUT (SO)**

This character (0/14) is used to invoke the G1 set into the in-use table (see D.4.3.2 and D.4.3.3).

#### **D.6.1.3.2 SHIFT-IN (SI)**

This character (0/15) is used to invoke the G0 set into the in-use table (see D.4.3.2 and D.4.3.3).

#### **D.6.1.3.3 SINGLE-SHIFT TWO (SS2)**

This character (1/9) is used to invoke the G2 set into the in-use table in a nonlocking manner (see D.4.3.2 and D.4.3.3).

#### **D.6.1.3.4 SINGLE-SHIFT THREE (SS3)**

This character (1/13) is used to invoke the G3 set into the in-use table in a non-locking manner (see D.4.3.2 and D.4.3.3).

#### **D.6.1.3.5 ESCAPE (ESC)**

This character (1/11) is used for code extension (see D.4.3.2 and D.4.3.3).

### **D.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), have no effect on the presentation layer and are reserved for use at other protocol layers. They may be embedded within any presentation layer sequence without affecting that sequence.

### **D.6.1.5 Device control characters**

The device control characters, i.e. DC1 (1/1), DC2 (1/2), DC3 (1/3), and DC4 (1/4), have no effect on the presentation layer and are reserved for use at other protocol layers. They may be embedded within any presentation layer sequence without affecting that sequence.

### **D.6.1.6 Other control characters**

#### **D.6.1.6.1 NULL (NUL)**

This character (0/0) has no effect on the presentation layer and is reserved for use at other protocol layers. It may be embedded within any presentation layer sequence without affecting that sequence.

#### **D.6.1.6.2 BELL (BEL)**

This character (0/7) is used to momentarily ring a bell or effect another transient indication.

### **D.6.1.6.3 CANCEL (CAN)**

This character (1/8) is used to terminate processing of all currently executing macros. Execution is resumed at the next presentation layer 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 layer code. The operation of the CAN character is not guaranteed unless it is guaranteed to be delivered by the lower layers.

### **D.6.1.6.4 SERVICE DELIMITER CHARACTER (SDC)**

This character (1/10) shall be executed as a null operation at the presentation layer and any other use is implementation-dependent (see Appendix IV).

### **D.6.1.6.5 NON-SELECTIVE RESET (NSR)**

This character (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:

- 1) The G0, G1, G2, G3, C0, and C1 sets are designated to their default states and the in-use code table is invoked to its default state, as described in D.4.3.
- 2) The DOMAIN parameters are set to their default values, as described in D.5.3.2.2.
- 3) The text parameters (from the TEXT opcode, from the C1 set and the active field), as described in D.5.3.2.9.3, are set to their default values.
- 4) The TEXTURE parameters are set to their default values, as described in D.5.3.2.4. The programmable masks are not cleared.
- 5) The colour mode is set to colour mode 0 and the drawing colour is set to nominal white. The colour map is not changed.
- 6) If the two bytes that immediately follow the NSR are both from columns 4 through 7 of the in-use table, the cursor is positioned. These two bytes represent, in binary form (i.e. the binary digit comprising the bits  $b_6$  through  $b_1$  with  $b_6$  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 in the display area and refer to the nominal screen format established by the default character size. The top of the character field for row 0 coincides with the top boundary of the display area. If either of the two bytes following the NSR is not from columns 4 through 7 (or columns 12 through 15) of the in-use table, the nonselective reset function of NSR is executed and the cursor is not repositioned. If the two bytes are from columns 2 and 3 (or columns 10 and 11), they are ignored. If either of the two bytes are C0 or C1 control characters (columns 0, 1 or 8, 9), they terminate the NSR sequence and they are executed.

## **D.6.2 C1 control set**

**D.6.2.1** The C1 control set (see Figure D.65) 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 D.6.2.2 to D.6.2.6.

### **D.6.2.2 Macros-General**

#### **D.6.2.2.1 DEF MACRO**

This C1 control is used to define a macro. Bits  $b_7$  through  $b_1$  of the character following this control shall be in the range 2/0 to 7/15 and are the bit combination representing the macro being defined. All characters subsequent to this character are stored (but not executed) within the receiving device 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, DEF TEXTURE, or END. Neither the terminating control character nor its preceding ESC character in a 7-bit environment is stored as part of the macro. If the character following the DEF MACRO control is not in this range, the entire command (i.e. the C1 control and the out of range character) is in error and is executed as a null operation.

b <sub>8</sub>	1	1				
b <sub>7</sub>	0	0				
b <sub>6</sub>	0	0				
b <sub>5</sub>	0	1				
	8	9				
b <sub>7</sub>	1	1				
b <sub>6</sub>	0	0				
b <sub>5</sub>	0	1				
	4	5				
b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	Column Row	A	B
0	0	0	0	00	DEF MACRO	PROTECT
0	0	0	1	01	DEFP MACRO	EDC <sub>1</sub>
0	0	1	0	02	DEFT MACRO	EDC <sub>2</sub>
0	0	1	1	03	DEF DRCS	EDC <sub>3</sub>
0	1	0	0	04	DEF TEXTURE	EDC <sub>4</sub>
0	1	0	1	05	END	WORD WRAP ON
0	1	1	0	06	REPEAT	WORD WRAP OFF
0	1	1	1	07	REPEAT TO EOL	SCROLL ON
1	0	0	0	08	REVERSE VIDEO	SCROLL OFF
1	0	0	1	09	NORMAL VIDEO	UNDER LINE START
1	0	1	0	10	SMALL TEXT	UNDER LINE STOP
1	0	1	1	11	MEDIUM TEXT	FLASH CURSOR
1	1	0	0	12	NORMAL TEXT	STEADY CURSOR
1	1	0	1	13	DOUBLE HEIGHT	CURSOR OFF
1	1	1	0	14	BLINK START	BLINK STOP
1	1	1	1	15	DOUBLE SIZE	UNPRO- TECT

T0823150-95/d259

#### 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 D.65/T.101 – C1 control set**

During the definition of a macro, a reference to a macro results in the storage of that reference only, not the expansion. The definition of a macro replaces any previously existing macro under the same name. A macro may be longer or shorter than the previously existing macro that it replaces. 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.

### **D.6.2.2.2 DEFP MACRO**

The operation of this control character is identical to that of the DEF MACRO command, except that incoming characters that make up the macro definition are simultaneously executed and stored. A macro is considered to be undefined during definition until the definition is terminated. Therefore, if a DEFP MACRO command contains a reference to itself, or if it references another macro which references the one being defined, the reference to the macro being defined is executed as a null operation.

### **D.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.

### **D.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. Bits  $b_7$  through  $b_1$  of the character following this control shall be in the range 2/0 to 7/15 and are the bit combination representing the DRCS character being defined. Next comes any valid stream of presentation layer code. The DRCS downloading command 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 terminated by another DEF DRCS command, the next character of the DRCS G-set (i.e. in the circular sequence 2/0, 2/1, ... 2/15, 3/0, 3/1 ... 7/14, 7/15, 2/0, 2/1 ...) is defined by the presentation layer 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 bits  $b_7$  through  $b_1$  of the character following the DEF DRCS control are not in the range 2/0 to 7/15 and the DEF DRCS control is not terminating a previous DEF DRCS command, the entire command (i.e. the C1 control and the out of range character) is in error and is executed as a null operation. If a DRCS definition is immediately terminated with no intervening presentation layer 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 layer code defining the DRCS character shall be executed upon being received. It is executed within the unit screen but is not displayed in the display area. The effect of this execution is to modify a separate storage buffer that is used for any subsequent display of the DRCS character. The effect on the physical display screen of the redefinition or deletion of a DRCS character that is being displayed is undefined.

All presentation layer code shall have the usual effect on the state of the receiving device with the single exception that all drawing operations affect the DRCS storage buffer rather than the display area. For example, if the colour map is changed, then any part of the physical display screen using that colour map entry will change colour. Any changes to the state of the receiving device, such as character field size, in-use G-sets, etc., shall persist after the completion of the DRCS definition.

The aspect ratio of the storage buffer shall be the same as that of the character field dimensions when the DEF DRCS character is received. The lower left corner of the buffer shall coincide with the lower left corner of the unit screen. The larger buffer dimension (dx or dy) shall coincide with the entire corresponding axis of the unit screen. For example, a character field that is 6/256 (of the unit screen) wide and 10/256 high would cause the DRCS character shape to be defined only by code that writes into that portion of the unit screen in the range 0.0 to 0.6 in the X axis and 0.0 (inclusive) to 1.0 (non-inclusive) in the Y axis. If the character field size is changed within the DRCS definition, it will have no effect on the aspect ratio and resolution of the storage buffer, but it will have an effect on the execution of subsequent characters within the unit screen.

The storage buffer for each DRCS character is divided into elements. Each element may be in either an off state or an on state. At the beginning of each DRCS definition, all of the elements in its storage buffer shall be set to the off state by the receiving device. The state of each element of the storage buffer shall be affected by presentation layer code that

writes into that portion of the unit screen which coincides with the element during the DRCS definition. Each element that is affected shall be set to the on state, unless it is written into with nominal black, in which case it is set to the off state. The effective resolution of the storage buffer is implementation dependent.

After the downloading sequence has been terminated, the receiving device reverts to the normal procedure of mapping the unit screen to the physical 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 of displaying a DRCS character shall be to scale the contents of its storage buffer to fit into the current character field. The elements in the on state shall be displayed in the drawing colour at the time of display, not in the drawing colour at the time of definition. In colour mode 2, the elements in the off state shall be displayed in the background colour. The display of DRCS characters shall be subject to all TEXT attributes. Note that if the current character field is neither the same size as, nor an integer multiple of, the character field size at the beginning of the DRCS definition, then some distortion may occur due to scaling and interpolation. Note also that a storage buffer as described here is not the only way to achieve the defined effects.

#### **D.6.2.4 DEF TEXTURE**

This command is used to define one of the four programmable texture masks described in D.5.3.2.4.

Bits  $b_7$  through  $b_1$  of the character following this control shall be one of the following bit combinations: (4/1), (4/2), (4/3), (4/4), that causes 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 layer code follows, it describes the texture mask in the same manner as DRCS characters, except that the texture mask size is used rather than the character field size. The DEF TEXTURE command is terminated when an END, DEF MACRO, DEFP MACRO, DEFT MACRO, DEF DRCS, or another DEF TEXTURE command is received. If bits  $b_7$  to  $b_1$  of the character following the DEF TEXTURE control are not in the range 4/1 to 4/4, the entire command (i.e., the C1 control and the out of range character) is in error and is executed as a null operation. At the end of the DEF TEXTURE command, the receiving device reverts to the normal procedure of mapping the unit screen to the physical display screen, with the drawing point reset to (0,0).

NOTE – The INCREMENTAL POINT command may scale the actual active field before execution (see D.5.3.3.6.3), causing the actual area defined to be smaller than requested.

#### **D.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 (see D.6.2.6).

#### **D.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. The method of entering and editing data into these fields is implementation-dependent.

By default, the entire unit screen is protected, i.e. it is not possible for the user to enter or alter data that displays on the unit screen. However, if the UNPROTECT command is given, the active field (as defined by the FIELD command described in D.5.3.3.6.2) is made unprotected and the user may subsequently enter or locally edit data within that field. Any number of unprotected fields may be defined (subject to implementation-dependent memory limitations) by defining an active field via a FIELD command and then issuing the UNPROTECT command. Should an UNPROTECT command result in unprotecting 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 field is made unprotected. This prevents unprotected fields from overlapping.

Data that are received and displayed in an unprotected field after it has been unprotected, and when that unprotected field coincides with the active field, can also be subsequently edited by the user. When the user initiates a transmission, the information in the unprotected field(s) is transmitted in conformance with this standard. 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, and then by the END command. When more than one unprotected field is transmitted, the order of

transmission is 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 leftmost first. The transmit presentation process and its associated state shall be maintained separately from the receive presentation process.

Unprotected fields may be reprotected via the PROTECT command. This command causes all unprotected fields of which any portion lies within the active field to be made protected. The RESET command may be used to protect all currently defined unprotected fields.

The use of unprotected fields does not preclude the use of other user input modes that are independent of and that do not affect the state of the unprotected fields. Such alternative input modes are implementation-dependent and may temporarily use the display area.

#### **D.6.2.7 Text controls**

**D.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.

##### **D.6.2.7.2 REPEAT**

This command causes the immediately preceding byte to be repeated a specific number of additional times if the byte is SPACE or any spacing character from the primary, supplementary, DRCS, or mosaic sets. Otherwise, the command is in error and shall be executed as a null operation. Bits  $b_6$  to  $b_1$  of the character following the REPEAT character shall be interpreted as the repeat count. Bits  $b_7$  through  $b_1$  of this repeat count character shall be in the range 4/0 through 7/15; otherwise the command is in error and shall be executed as a null operation, and the count character is executed as a character from columns 0 to 3 or 8 to 11.

##### **D.6.2.7.3 REPEAT TO EOL**

This command causes the immediately preceding byte to be repeated until the last character position along the current character path is reached if the byte is SPACE or any spacing character from the primary, supplementary, DRCS, or mosaic sets. Otherwise the command is in error and shall be executed as a null operation. If the full character field corresponding to the text cursor lies entirely within the active field when this command is encountered, then characters are repeated only up to the last character position along the current character path within the active field.

##### **D.6.2.7.4 REVERSE VIDEO**

This command causes the receiving device to enter reverse video mode in which any subsequently received alphanumeric text, mosaic, and DRCS characters are drawn so that the pixels surrounding the character shape in the character field are drawn in the drawing colour. The pixels of the character shape are not drawn, except in colour mode 2, when they are drawn in the background colour.

##### **D.6.2.7.5 NORMAL VIDEO**

This command causes the receiving device to exit reverse video mode. This is the default state.

##### **D.6.2.7.6 SMALL TEXT**

This command causes the dimensions of the character field to be set to  $dx = 1/80$  and  $dy = 5/128$ , consistent with the physical resolution.

##### **D.6.2.7.7 MEDIUM TEXT**

This command causes the dimensions of the character field to be set to  $dx = 1/32$  and  $dy = 3/64$ , consistent with the physical resolution.

##### **D.6.2.7.8 NORMAL TEXT**

This command causes the dimensions of the character field to be set to their default value; that is,  $dx = 1/40$  and  $dy = 5/128$ , consistent with the physical resolution.

##### **D.6.2.7.9 DOUBLE HEIGHT**

This command causes the dimensions of the character field to be set to  $dx = 1/40$  and  $dy = 5/64$ , consistent with the physical resolution.

#### **D.6.2.7.10 DOUBLE SIZE**

This command causes the dimensions of the character field to be set to  $dx = 1/20$  and  $dy = 5/64$ , consistent with the physical resolution.

Note that SMALL TEXT, MEDIUM TEXT, NORMAL TEXT, DOUBLE HEIGHT, and DOUBLE SIZE only affect the sign and magnitude of the character field dimensions. They do not affect rotation, character path, intercharacter and interrow spacing, reverse video, scroll, word wrap, underline, cursor state, cursor style, or move attributes.

#### **D.6.2.7.11 WORD WRAP ON**

This command causes the receiving device to enter word wrap mode. In this mode, subsequently received alphanumeric text is buffered into words. A word is displayed on the current line only if the entire buffered word will fit into the space remaining on the current line within the unit screen (or within the active field, should the full character field corresponding to the text cursor lie entirely within the active field). If the word does not fit into the space remaining on the current line, an automatic APR APD is executed 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 clause, a word is defined as being an accumulation of characters between SPACE characters.

Words consisting entirely of alphabetic characters (see Table D.18) 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 shall 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.

#### **D.6.2.7.12 WORD WRAP OFF**

This command causes the receiving device to exit word wrap mode. In this mode (the default mode), all text is broken on character boundaries whenever an automatic APR and APD are executed.

#### **D.6.2.7.13 SCROLL ON**

In this mode, a subsequently received APD or APU command or an automatic APR APD that would advance any part of the cursor out of the display area (or the active field, should the full character field corresponding to the cursor lie entirely within the active field) causes the entire display within the area or field 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 or field. The colour of background pixels scrolled into the area or field is nominal black in colour modes 0 and 1 and the background colour in colour mode 2.

Buffering of data scrolled out of the display area or active field is implementation-dependent.

#### **D.6.2.7.14 SCROLL OFF**

In this mode (the default mode) an APD or APU command or an automatic APR APD that would advance any part of the cursor out of the display area (or the active field, should the full character field corresponding to the cursor lie entirely within the active field), causes the cursor to be repositioned to the opposite edge of the area or field such that the character field so defined lies entirely within the area or field.

#### **D.6.2.7.15 UNDERLINE START**

This command causes the receiving device to enter underline mode. In this mode, all subsequently received primary, supplementary, DRCS characters, and the SPACE character have a line added to their patterns. The line appears in the character field starting at the character field origin and extending across the entire width ( $dx$  dimension) of the character field, but not across the space between character fields when the intercharacter spacing is greater than one. Its thickness is determined by the vertical dimension ( $dy$ ) of the logical pel size. The underline mode also causes subsequently received mosaic characters to be displayed in separated mode (see D.5.4). Mosaic characters are not underlined.

#### **D.6.2.7.16 UNDERLINE STOP**

This command causes the receiving device 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.

#### **D.6.2.7.17 FLASH CURSOR**

This command turns on the cursor display and causes it to flash intermittently and may enable user input (see D.6.2.6) and user activation of linked macros (see D.5.5).

#### **D.6.2.7.18 STEADY CURSOR**

This command turns on the cursor display, which remains constantly visible and may enable user input (see D.6.2.6) and user activation of linked macros (see D.5.5).

#### **D.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. Whether CURSOR OFF disables or buffers or has no effect on user input (see D.6.2.6) and/or disables or has no effect on user activation of linked macros (see D.5.5) is implementation-dependent.

### **D.6.2.8 Other controls**

#### **D.6.2.8.1 BLINK START**

This command creates a blink process in which: the blink-from colour is the drawing colour; the blink-to colour is nominal black in colour modes 0 and 1 or the background colour in colour mode 2; the on and off intervals are implementation-dependent; and the phase delay is 0.

In colour mode 0 the C1 BLINK START command establishes an automatic process that implicitly defines a blinking colour. Note that objects drawn with the previous drawing colour will remain not blinking, but objects subsequently drawn with the new drawing colour will blink. If the drawing colour is changed, the old colour remains blinking and the new drawing colour does not blink.

#### **D.6.2.8.2 BLINK STOP**

This command (the default condition) turns off any currently active blink processes utilizing the drawing colour as the blink-from colour.

#### **D.6.2.8.3 EXTENDED DEVICE CONTROLS (EDC1, EDC2, EDC3, and EDC4)**

The precise meanings of EDC1, EDC2, EDC3, and EDC4 are reserved for future standardization, and are executed as null operations.

## **D.7 Graphic character repertoire**

**D.7.1** The repertoire of graphic characters and their coded representations are specified by Tables D.18 to D.26 and the characters of Table D.27 when combined with SPACE. The non-spacing characters of Table D.27 may be used in combination with any graphic character of the repertoire, including an accented character. However, such combinations or other combinations (such as using APB or other positioning commands) do not constitute characters of the repertoire. Any character of the graphic character repertoire shall be displayed in all colour modes. This may require special attention in colour mode 2 and in reverse video mode. The presentation of characters that are not in the graphic character repertoire is implementation dependent.

**D.7.2** In Tables D.18 to D.27, the Coded Representation column specifies the coded representation of the graphic characters for the 7-bit and 8-bit environments. The symbol S is used to identify that the immediately following bit combination represents an element of the supplementary set. In the tables, the coded representations without the symbol S refer to characters in the primary set. Where the coded representation consists of 2-bit combinations (e.g. lower case a with acute accent: S 4/2 6/1), the left bit combination shall precede the right one in the information interchange. 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. In the 7-bit environment, elements of the supplementary set shall be represented by SS2 or SS3 followed by 2/1 to 7/14 when the supplementary set is designated as G2 or G3, respectively. In the 8-bit environment, elements of the supplementary set shall be represented either by SS2 followed by 2/1 to 7/14 when the supplementary set is designated as G2 or as 10/1 to 15/14 when the supplementary set is invoked into GR. The actual number of bit combinations needed to code accented characters depends on the invocation sequence required (if any) to invoke the supplementary and primary sets. The invocation sequence for the primary set may occur after the supplementary character, in the coding of accented characters. This character repertoire is based on CCITT Recommendation V.3 (1972), ISO 646 (1983), CCITT S.100 (1980), and ISO 6937 (1982).

**D.7.3** In Tables D.18 to D.27, the coded representations merely refer to the primary set and supplementary set without reference as to which G-sets these two sets are designated and invoked into. In this Data Syntax, the primary set can be designated and invoked as a G0, G1, G2, or G3 set, with G0 as the default and the supplementary set can be designated and invoked as G0, G1, G2 or G3 set, with G2 as default

**Table D.18/T.101 – 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
Ḃ	Lower case a with breve	S 4/6 6/1
Ḃ	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
æ	Lower case æ diphthong	S 7/1
Æ	Upper case Æ diphthong	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
ç	Lower case c with acute accent	S 4/2 6/3
Ç	Upper case C with acute accent	S 4/2 4/3

**Table D.18/T.101 – Latin alphabetic characters** (*continued*)

Graphic	Name or description	Coded representation
·	Lower case c with circumflex accent	S 4/3 6/3
◌̂	Upper case C with circumflex accent	S 4/3 4/3
◌̣	Lower case c with caron	S 4/15 6/3
◌̤	Upper case C with caron	S 4/15 4/3
◌̇	Lower case c with dot	S 4/7 6/3
◌̈	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
◌̣ or 'd	Lower case d with caron	S 4/15 6/4
◌̤	Upper case D with caron	S 4/15 4/4
◌̆	Lower case d with stroke	S 7/2
◌̇	Upper case D with stroke, Icelandic eth	S 6/2
☆	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
◌̣	Lower case e with caron	S 4/15 6/5
◌̤	Upper case E with caron	S 4/15 4/5
◌̇	Lower case e with dot	S 4/7 6/5
◌̈	Upper case E with dot	S 4/7 4/5
◌̄	Lower case e with macron	S 4/5 6/5
◌̅	Upper case E with macron	S 4/5 5/5

**Table D.18/T.101 – Latin alphabetic characters (continued)**

Graphic	Name or description	Coded representation
◌̡	Lower case e with ogonek	S 4/14 6/5
◌̢	Upper case E with ogonek	S 4/14 4/5
f	Lower case f	6/6
F	Upper case F	4/6
g	Lower case g	6/7
G	Upper case G	4/7
◌̣	Lower case g with acute accent	S 4/2 6/7
◌̤	Lower case g with circumflex accent	S 4/3 6/7
◌̥	Upper case G with circumflex accent	S 4/3 4/7
◌̦	Lower case g with breve	S 4/6 6/7
◌̧	Upper case G with, breve	S 4/6 4/7
◌̨	Lower case g with dot	S 4/7 6/7
◌̩	Upper case G with dot	S 4/7 4/7
◌̪	Upper case G with cedilla	S 4/11 4/7
h	Lower case h	6/8
H	Upper case H	4/8
◌̫	Lower case h with circumflex accent	S 4/3 6/8
◌̬	Upper case H with circumflex accent	S 4/3 4/8
◌̭	Lower case h with stroke	S 7/4
◌̮	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
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

**Table D.18/T.101 – Latin alphabetic characters** (*continued*)

Graphic	Name or description	Coded representation
◌̃	Lower case i with tilde	S 4/4 6/9
Ɑ	Upper case I with tilde	S 4/4 4/9
◌̇	Upper case I with dot	S 4/7 4/9
◌̄	Lower case i with macron	S 4/5 6/9
Ɱ	Upper case I with macron	S 4/5 4/9
◌̇̈	Lower case i with ogonek	S 4/14 6/9
Ɐ	Upper case I with ogonek	S 4/14 4/9
ɿ	Lower case ij ligature	S 7/6
ⱪ	Upper case IJ ligature	S 6/6
◌̈	Lower case i without dot	S 7/5
j	Lower case j	6/10
J	Upper case J	4/10
Ⱬ	Lower case j with circumflex accent	S 4/3 6/10
ⱬ	Upper case J with circumflex accent	S 4/3 4/10
k	Lower case k	6/11
K	Upper case K	4/11
Ɀ	Lower case k with cedilla	S 4/11 6/11
Ȿ	Upper case K with cedilla	S 4/11 4/11
Ⱬ	Lower case k, Greenlandic	S 7/0
l	Lower case l	6/12
L	Upper case L	4/12
◌́	Lower case l with acute accent	S 4/2 6/12
◌̇	Upper case L with acute accent	S 4/2 4/12
◌̂ or 'l	Lower case l with caron	S 4/15 6/12
◌̂ or 'L	Upper case L with caron	S 4/15 4/12
◌̈́	Lower case l with cedilla	S 4/11 6/12
◌̇̈́	Upper case L with cedilla	S 4/11 4/12
◌̣	Lower case l with stroke	S 7/8
◌̣	Upper case L with stroke	S 6/8

**Table D.18/T.101 – Latin alphabetic characters** (*continued*)

Graphic	Name or description	Coded representation
ł	Lower case l with middle dot	S 7/7
Ł	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
ñ	Lower case n with acute accent	S 4/2 6/14
Ñ	Upper case N with acute accent	S 4/2 4/14
ñ	Lower case n with tilde	S 4/4 6/14
Ñ	Upper case N with tilde	S 4/4 4/14
ɲ	Lower case n with caron	S 4/15 6/14
Ň	Upper case N with caron	S 4/15 4/14
ñ	Lower case n with cedilla	S 4/11 6/14
Ñ	Upper case N with cedilla	S 4/11 4/14
ɳ	Lower case eng, Lapp	S 7/14
Ń	Upper case eng, Lapp	S 6/14
ˆn	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

**Table D.18/T.101 – Latin alphabetic characters** (*continued*)

Graphic	Name or description	Coded representation
◌̈́	Lower case o with double acute accent	S 4/13 6/15
◌̈́	Upper case O with double acute accent	S 4/13 4/15
◌̄	Lower case o with macron	S 4/5 6/15
◌̄	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
◌̄	Lower case r with acute accent	S 4/2 7/2
◌̄	Upper case R with acute accent	S 4/2 5/2
◌̂	Lower case r with caron	S 4/15 7/2
◌̂	Upper case R with caron	S 4/15 5/2
◌̃	Lower case r with cedilla	S 4/11 7/2
◌̃	Upper case R with cedilla	S 4/11 5/2
s	Lower case s	7/3
S	Upper case S	5/3
◌̄	Lower case s with acute accent	S 4/2 7/3
◌̄	Upper case S with acute accent	S 4/2 5/3
◌̂	Lower case s with circumflex accent	S 4/3 7/3
◌̂	Upper case S with circumflex accent	S 4/3 5/3
◌̂	Lower case s with caron	S 4/15 7/3
◌̂	Upper case S with caron	S 4/15 5/3
◌̃	Lower case s with cedilla	S 4/11 7/3
◌̃	Upper case S with cedilla	S 4/11 5/3

**Table D.18/T.101 – Latin alphabetic characters (continued)**

Graphic	Name or description	Coded representation
ß	Lower case sharp s, German	S 7/11
t	Lower case t	7/4
T	Upper case T	5/4
ŧ or 't	Lower case t with caron	S 4/15 7/4
Ť	Upper case T with caron	S 4/15 5/4
ƚ	Lower case t with cedilla	S 4/11 7/4
Ƨ	Upper case T with cedilla	S 4/11 5/4
ƚ̣	Lower case t with stroke	S 7/13
Ƨ̣	Upper case T with stroke	S 6/13
ƚ̨	Lower case thorn, Icelandic	S 7/12
Ƨ̨	Upper case thorn, Icelandic	S 6/12
u	Lower case u	7/5
U	Upper case U	5/5
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
Û	Upper case U with diaeresis or umlaut mark	S 4/8 5/5
ũ	Lower case u with tilde	S 4/4 7/5
Û	Upper case U with tilde	S 4/4 5/5
ū	Lower case u with breve	S 4/6 7/5
Û	Upper case U with breve	S 4/6 5/5
•	Lower case u with double acute accent	S 4/13 7/5
◌̧	Upper case U with double acute accent	S 4/13 5/5
ũ	Lower case u with ring	S 4/10 7/5
Û	Upper case U with ring	S 4/10 5/5

**Table D.18/T.101 – Latin alphabetic characters (end)**

Graphic	Name or description	Coded representation
·	Lower case u with macron	S 4/5 7/5
Ɑ	Upper case U with macron	S 4/5 5/5
ł	Lower case u with ogonek	S 4/14 7/5
Ł	Upper case U with ogonek	S 4/14 5/5
v	lower case v	7/6
V	Upper case V	5/6
w	Lower case w	7/7
W	Upper case W	5/7
•	Lower case w with circumflex accent	S 4/3 7/7
ⱦ	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
Ʒ	Lower case y with acute accent	S 4/2 7/9
Ÿ	Upper case Y with acute accent	S 4/2 5/9
ÿ	Lower case y with circumflex accent	S 4/3 7/9
Ÿ	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
Ʒ	Lower case z with acute accent	S 4/2 7/10
Ʒ	Upper case Z with acute accent	S 4/2 5/10
Ʒ	Lower case z with caron	S 4/15 7/10
Ʒ	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 D.19/T.101 – 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 D.20/T.101 – 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 D.21/T.101 – 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 D.22/T.101 – 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
×	Multiply sign	S 3/4

**Table D.23/T.101 – 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 D.24/T.101 – 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 D.25/T.101 – 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
®	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

**Table D.25/T.101 – Miscellaneous symbols (end)**

Graphic	Name or description	Coded representation
	Diagonal (Note 1)	S 5/8
	Reverse diagonal (Note 2)	S 5/9
	Filled diagonal (Note 3)	S 5/10
	Filled reverse diagonal (Note 4)	S 5/11
	Cross (Note 5)	S 6/5
	Full horizontal line (Note 6)	S 5/6
	Full vertical line (Note 7)	S 5/7
–	Horizontal bar (Note 8)	S 5/0
<p><b>NOTES</b></p> <p>1 The diagonal extends from the lower left corner of the character field to the upper right corner of the character field.</p> <p>2 The reverse diagonal extends from the lower right corner of the character field to the upper left corner of the character field.</p> <p>3 The filled diagonal is a fitted triangle occupying the lower right half of the character field.</p> <p>4 The filled reverse diagonal is a filled triangle occupying the lower left half of the character field.</p> <p>5 The cross character is equivalent to overlaying the full horizontal line character with the full vertical line character.</p> <p>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.</p> <p>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.</p> <p>8 The typical graphic representation of the horizontal bar is a horizontal, line about level with the middle of a capital letter.</p> <p>9 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.</p>		

**Table D.26/T.101 – 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
◌̣	Caron	S 4/15 2/0
ˆˆ	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
<p>NOTE – The grave accent, circumflex, and tilde marks are also coded as 6/0, 5/14, and 7/14 respectively.</p>		

**Table D.27/T.101 – Miscellaneous non-spacing characters**

Graphic	Name or description	Code representation
□	Non-spacing underline	S 4/12
—	Non-spacing vector overbar	S 4/10
/	Non-spacing slant	S 4/9
<p>NOTES</p> <p>1 The graphic representation of non-spacing vector overbar is that of a vector arrow at just above the top of a capital letter. When non-spacing underline, non-spacing vector overbar, and non-spacing slant are used, their coded representations precede those of the characters in Tables D.18 through D.26.</p> <p>2 The rectangle surrounding the non-spacing underline is for illustrative purposes only and is not part of the graphic symbol.</p>		

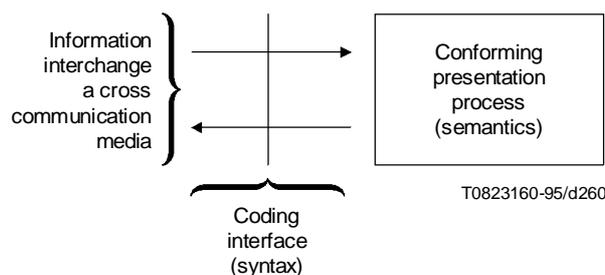
## D.8 Conformance requirements

### D.8.1 General

This Data Syntax provides a coding scheme by which specific information may be conveyed and correctly interpreted regardless of the hardware dependencies of particular receiving device configurations. This provides manufacturers and information providers with the flexibility to develop a range of products and service offerings to address different segments of the market. However, the existence of ranges of service causes problems both for the manufacturer and for the information provider. Receiving display device manufacturers may wish to interpret and display only the information that their products can handle, whereas information providers may wish all receiving devices to display all the information in a consistent manner. 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 in both directions across the coding interface, it is essential that all interchange be in full compliance with the syntactical rules as defined in this Data Syntax (see Figure D.66). 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.

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's). A general SRM for videotex and a general SRM for teletext are in Appendix IV and should serve as a guideline in defining additional SRM's.



**Figure D.66/T.101 – Relationship of conforming entities**

## D.8.2 Conforming interchange

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

- 1) Shall be a sequence of 7-bit or 8-bit bit combinations following the syntactic formats described throughout the body of this Data Syntax.
- 2) Shall not include any byte or value reserved for future standardization. This is intended to facilitate forward compatibility with any future versions of this Data Syntax.
- 3) Shall not include any bit combinations allocated by this Data Syntax for any purpose other than that defined in this Data Syntax, unless they represent functions or elements of other code sets that have been invoked by code extension according to ISO 2022-1982, subject to mutual agreement between interchange parties.
- 4) Shall not include sequences that would result in an undefined presentation process or those which are indicated as errors in this Data Syntax; for example, a drawing point shall not be placed outside the unit screen.
- 5) Shall immediately follow sequences that result in an implementation dependent state of the presentation process, with those sequences that will reestablish the presentation process to a known state. For example, the positioning with respect to automatic APR APD, word wrap, and proportional spacing is implementation-dependent. Conforming interchange shall reestablish the position of the cursor and the graphic drawing point if tied to the cursor.

## D.8.3 Conforming Presentation Process

**D.8.3.1** Hereafter, the following definitions apply:

**D.8.3.1.1** **recognize** means to determine the syntactic form of a code sequence.

**D.8.3.1.2** **execute** means to process a code sequence to allow the display of information conveyed by the code sequence, if presented, and by subsequent code sequences, as specified in this Data Syntax (e.g. computation of positional information).

**D.8.3.1.3** **present** means to display the information conveyed by a code sequence, and, in the case of a control function, to display subsequent information affected by the control function.

**D.8.3.2** A receive presentation process conforming to this Data Syntax shall be capable of receiving all interchange and recognizing whether or not it is conforming (as defined in D.8.2). Non-conforming interchange shall be ignored, unless otherwise specified in this Data Syntax. This is intended to facilitate backward compatibility with any future versions of this Data Syntax.

**D.8.3.3** A receiving display device conforming to this Data Syntax:

- 1) Shall execute all code extension sequences defined in this Data Syntax.
- 2) Shall execute and present all 94 characters of the primary set consistent with the text attributes implemented.
- 3) Shall execute all remaining characters of the graphic character repertoire (see D.7.1) and present them exactly or in an approximate form or as a symbol(s) that cannot be confused with any character in the graphic character repertoire.
- 4) Shall execute and present all geometric graphic primitives (i.e. POINT, LINE, ARC, RECTANGLE, POLYGON, and INCREMENTAL), shall execute and present the control function RESET, and shall execute the control functions DOMAIN and WAIT.
- 5) Shall execute all discrete values of the attribute control functions which affect the recognition and execution of subsequent code sequences, i.e. singlevalue operand length (1 to 4), multi-value operand length (1 to 8), dimensionality (2, 3), and colour modes (0, 1, and 2).

- 6) Shall execute all other control codes from the PDI set and shall present them consistently with the attributes implemented.
- 7) Shall execute and present all 65 codes of the mosaic set
- 8) Shall execute and present all 96 macros, which are defined according to the procedures of this Data Syntax.
- 9) Shall execute and present all 96 DRCS characters, which are dynamically redefinable according to the procedure of this Data Syntax.
- 10) Shall execute and present all C0 control functions defined in this Data Syntax.
- 11) Shall execute all control codes from the CI code set as defined in this Data Syntax, except for one-way services in which the following may not be required: FLASH CURSOR, STEADY CURSOR, CURSOR OFF, PROTECT, and UNPROTECT. The presentation of the CI control codes is implementation-dependent except for REPEAT, REPEAT TO EOL, SCROLL ON, and SCROLL OFF, which shall be executed and presented.

**D.8.3.4** In a receiving display device conforming to this Data Syntax, attributes may be implemented to a variety of capabilities, as follows:

- 1) All parameters for drawing commands or attributes that are specified as continuous variables may be approximated with an implementation-dependent value or values, one value of which shall be the default value.
- 2) All parameters for drawing commands or attributes that may take on multiple (two or more) discrete values may be approximated with an implementation-dependent value or values, one value of which shall be the default value.
- 3) 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.

**D.8.3.5** The default conditions of the attributes for all transmit and receive presentation processes conforming to this Data Syntax are summarized in Table D.28.

## **D.9 Enhanced Capabilities**

The future addition of enhanced capabilities is accommodated by the code extension techniques defined in this Data Syntax. 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 C- and G-sets described herein beyond that already defined, for example, to PDI commands which are specified with fixed sequences or in commands indicated to be in error where the receiving device is to ignore the entire command, is reserved for future compatible extension to this Data Syntax.

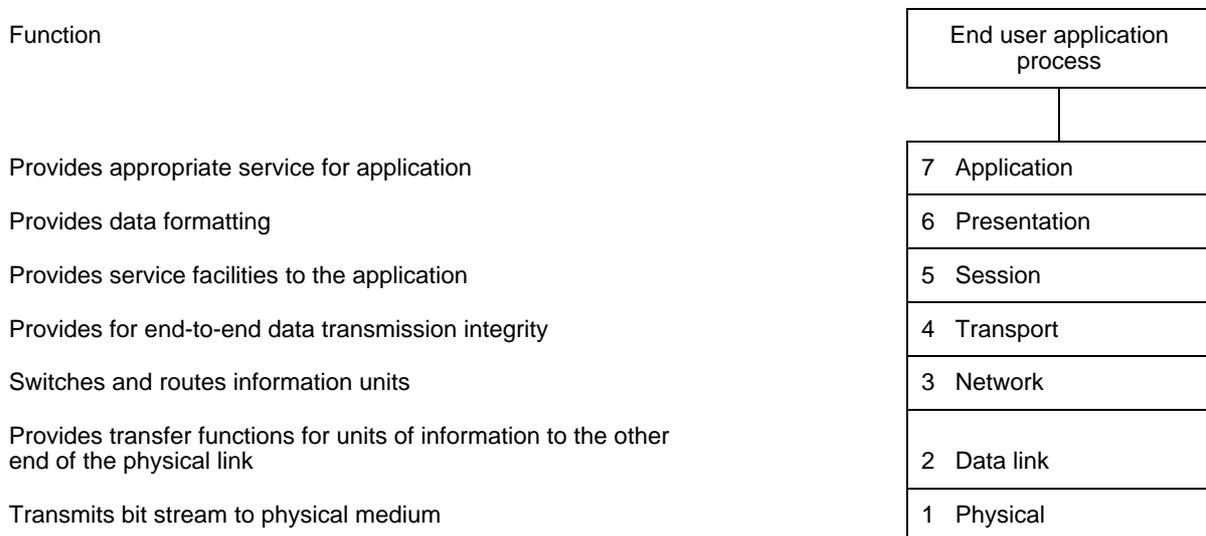
**Table D.28/T.101 – Default conditions**

Parameters	Values
<b>a) Default condition on initialization and as re-established by NSR</b>	
In-use table and active C- and G-sets:	As shown in Figures D.4 and D.6
Colour mode:	0
Drawing colour:	White
Single-value length (colour map address):	1 byte
Multi-value length (coordinate data):	3 bytes
Dimensionality:	2 dimensions with Z = 0
Logical pel size:	dx = 0, dy = 0 (origin at lower left)
Text rotation:	0 degrees (relative to horizontal)
Character path:	To the right
Intercharacter spacing:	1 (width of current character field)
Interrow spacing:	1 (height of current character field)
Move parameters:	Cursor and graphics drawing point move together
Cursor and drawing point positions:	Undefined
Cursor style:	Underscore
Cursor display:	Off (invisible)
Character field dimensions:	dx = 1/40, dy = 5/128, consistent with the physical resolution
Line texture:	Solid
Highlight:	No highlight
Texture pattern:	Solid
Texture mask size:	dx = 1/40, dy = 5/128, consistent with the physical resolution
Active field:	The unit screen
Underline mode:	Off
Word wrap mode:	Off
Scroll mode:	Off
Reverse video mode:	Off
<b>b) Other default conditions on initialization</b>	
Colour map:	Half grey scale, half saturated hues, as defined in D.5.3.2.5.2
Blink condition:	Off
Macros:	Null
DRCS:	Null
Programmable texture patterns:	Null
Unprotected fields:	None defined

**Appendix I**  
(to Recommendation T.101, Annex D)

**Layered Architecture Model**

The Basic Reference Model for Open Systems Interconnection is described in ISO DIS 7498-1983. A similar concept is also defined in ITU-T Recommendation X.200. The layered system architecture is an assembly of interrelated protocols required to define an entire communication 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. Figure A.1 illustrates the model. The coding scheme described in this Data Syntax addresses itself to the user data conveyed by the presentation layer of the Basic Reference Model, and is independent of the protocols used to transfer data between systems.



**Figure I.1/T.101 – The seven functionally separate layers of the OSI model**

Both videotex and teletext services make use of the same presentation layer protocol data syntax and semantics but may have different protocols at other layers. The protocols for other layers for teletext are defined in CCIR Report 957, Characteristics of Teletext Systems, Document 11/5001-E, October 1981; Broadcast Specification BS-14, Department of Communications, Canada, Telecommunications Regulatory Service, June 1981; and the North American Basic Teletext Specification (NABTS).

The seven layers may be viewed in two major groupings. Layers 1 to 4 treat the transference of data while layers 5 to 7 treat 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 information in a data coding format in a way that preserves its meaning. The detailed coding formats for the scheme described in this appendix provide the basis of a presentation layer protocol data syntax 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.

## **Appendix II**

(to Recommendation T.101, Annex D)

### **Coordinate System Concepts**

This Data Syntax makes use of the concept of the abstract Cartesian coordinate system upon which all pictorial, textual, and incremental drawing operations are defined. Normalized unit coordinates are used. In two dimensions, the drawing plane consists of a square area, the unit screen, whose points along the X and Y axes range from 0 (inclusive) to 1 (non-inclusive). In three dimensions, the drawing space 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.

Coordinate specifications may be described to several levels of accuracy due to their representations as fractions in this Data Syntax. These fractions are written in the abstract terms of the unit drawing area only, in order not to eliminate any possible implementations. However, these “natural” fractions may not be exactly representable in the receiving device’s “internal” arithmetic, so some approximation is necessary. (Example:  $1/40$  is an irrational fraction in binary representation.) This is done by eliminating unrepresentable least significant bits by an implementation-dependent truncation process. One way to achieve “consistent with the physical resolution” for text and graphics, when a physical resolution of 256 pixels is specified in the SRM, is to maintain at least 12 bits of precision in the receiving device. For example, the NORMAL TEXT character width of  $1/40$  is represented as  $102/4096$  at 12 bits of precision. Note that for higher physical resolutions, correspondingly greater internal precision may be needed to maintain consistency with the physical resolution.

Conceptually there are several numerical representations in use in this communications process. First of all there is the theoretical or “natural” infinite precision representation in which pictures are defined. This must be represented within the limits of the syntax; that is, with a specified DOMAIN for communications.

Within a receiving device, this numerical representation is mapped to the “internal” precision of the device. Some precision may be lost at this stage. Coordinates (or positions) are maintained in this internal resolution. Actual drawing makes use of the physical display resolution. There is a mapping from the internal representation to the physical representation with another possible loss of precision.

Those physical pixels are illuminated to which any portion of the internal representation of the drawing maps; for example, an internal representation of 12 bits accuracy might map to a physical resolution of 256 pixels (8 bits). The mapping in this case would simply be a truncation of the low order 4 bits. If the representations are not simply related by a power of 2, then the mapping is more complex.

This Data Syntax purposely does not specify relationships between the three representations, only that they must be consistent. This is to allow differing implementations to achieve the consistency specified in the SRM.

This Data Syntax is written in the abstract terms of the unit screen 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.

This Data Syntax indicates that the exact registration of the pictorial, textual, and photographic drawing techniques is only guaranteed to within certain limits. This is because, for certain display resolutions or when using certain display technologies, 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 moir patterns in optics, which may cause a severe degradation of a picture.

Jagged edges similar to those which occur in drawing line segments may occur in character drawing or elsewhere, primarily due to scaling transformations. This may be eliminated or minimized by various implementation techniques.

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 degree line is smooth. Pixels must be square or very nearly square for there to be no appearance of geometric distortion. Circles must appear round. This means that in the vertical direction the physical pixels should align with the scan lines of the television display.

In the videotex and teletext SRM's, the default text size results in 40 characters per row and 20 rows guaranteed within the display area of a television display screen. The aspect ratio applies to the outer bounds of the television physical display screen. If a separate decoder and television display are used, then some allowance should be made in decoder design for overscan. See Society of Motion Picture and Television Engineers (SMPTE) Recommended Practice, Specifications for Safe Action and Safe Title Areas, Test Pattern for Television Systems, SMPTE RP 27.3-1972.

The number 40 does not divide exactly into 256 and, when a character field width 1/40 of the unit screen is defined, there may be quantization error effects. For example, the 40 character positions might be defined to fit on the display screen in slightly fewer than 256 pixel elements in order to minimize quantization error effects.

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.

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 or mask size that is greater than the physical limit implied by the memory size available.

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

- 1) 256 pixels in the X axis by 200 pixels in the Y axis;
- 2) an aspect ratio of 0.78125;
- 3) default character field physical dimensions of 102/4096 in X by 10/256 in Y.

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 102/4096 in X by 8/256 in Y.

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 and may result in quantization error.

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 III (to Recommendation T.101, Annex D)

#### Code extension and 7-bit/8-bit transform

ISO 2022 (1986), Information Processing-7-bit and 8-bit coded character sets-code extension techniques, specifies the structure of 7-bit and 8-bit coded character environments (see D.4.3). Code extension refers to the means to represent more graphic characters and control functions than 128 or 256 in 7 bits or 8 bits, respectively. There are two means of extension:

- 1) changing the interpretation of a C- or G-set by invoking a different set; and
- 2) representing a graphic character or control function as a sequence of bytes.

The first character of such a sequence is called an introducer since it alters the interpretation of following bytes. Each introducer has a specified syntax for following bytes.

In this Data Syntax, the sequences listed in Table III.1 are used to represent graphic characters or functions.

**Table III.1/T.101 – Introducers**

Name of introducer	Syntax examples	Reference
ESCAPE	ESC I ... IF	D.4.3.2
SS2, SS3	SS2 G	D.4.3.2
Non-spacing accent	G G	D.5.2
PDI opcode	P D ... D	D.5.3
APS	APS G G	D.6.1.2.4
NSR	NSR G G	D.6.1.6.5
DEF MACRO, DEFP MACRO, DEFT MACRO, DEF DRCS, DEF TEXTURE	DEF MACRO M ... END	D.6.2.2
REPEAT	REPEAT G	D.6.2.7.2

This Data Syntax also specifies an error recovery if an unexpected character is received in the sequence. The recovery is to treat the partial sequence as a null operation and to execute the offending character.

ISO 2022 (1986) specifies an algorithm to transform between 7-bit and 8-bit environments while making minimal assumptions about the coded character sets being transformed. Such transformations are typically performed at gateways between services. The assumptions are:

- 1) C-sets are columns 0-1 and 8-9.
- 2) G-sets are columns 2-7 and 10-15.
- 3) The C0 set contains ESC, SO, and SI coded as 1/11, 0/14, and 0/15, respectively.
- 4) Additional locking shifts are coded as ESC 6/14, ESC 6/15, ESC 7/14, ESC 7/13, and ESC 7/12.
- 5) Any other C0 or C1 character may be an introducer.

The simplest 7-bit to 8-bit transform merely adds  $b_8 = 0$  to the data. This means that GR is unused and C1 controls are represented as 2-byte ESC  $F_e$  sequences.

Consider the following example following the ISO 2022 transform:

7 bits: SI 4/1 SO 2/1 SS2 3/1

When transformed to 8 bits that puts G0 in GL and G1 in GR:

8 bits: 4/1 10/1 SS2 11/1

The byte following the SS2 has  $b_8$  set to 1, since the data stream is still in the shift out (G1) state. (The transform algorithm doesn't know about SS2 or any other introducers except ESC.) This is why the standard specifies that the 8th bit,  $b_8$ , is to be ignored in operand bytes following an introducer.

NOTE – For the bytes following the single operand byte after DEF's up to the END, bit  $b_8$  is significant.

Consider the following example where G0 is in GL and G1 is in GR:

8 bits: 4/1 10/1 SS2 3/1

When transformed to 7 bits:

7 bits: 4/1 SO 2/1 SS2 SI 3/1

The transform algorithm puts an SI before the 3/1 since the 8-bit data stream went from G1 (GR) to G0 (GL). Thus, a locking shift was introduced between the introducer and its operand byte. The developers of the standard consider it too complex to require a receiving device to be able to handle locking shifts between an introducer and its one or two bytes of operand data, especially since many of the locking shifts are 2-byte escape sequences themselves.

A preferred 8-bit to 7-bit transform algorithm would be to treat all C0 and C1 characters as if they were introducers and to always insert a locking shift before any C-set character followed by a G-set character that required a shift. This does require that the 8-bit to 7-bit transform algorithms buffer one character ahead.

## Appendix IV

(to Recommendation T.101, Annex D)

### A general Service Reference Model (SRM) for Videotex and a general Service Reference Model (SRM) for Teletext

#### Scope

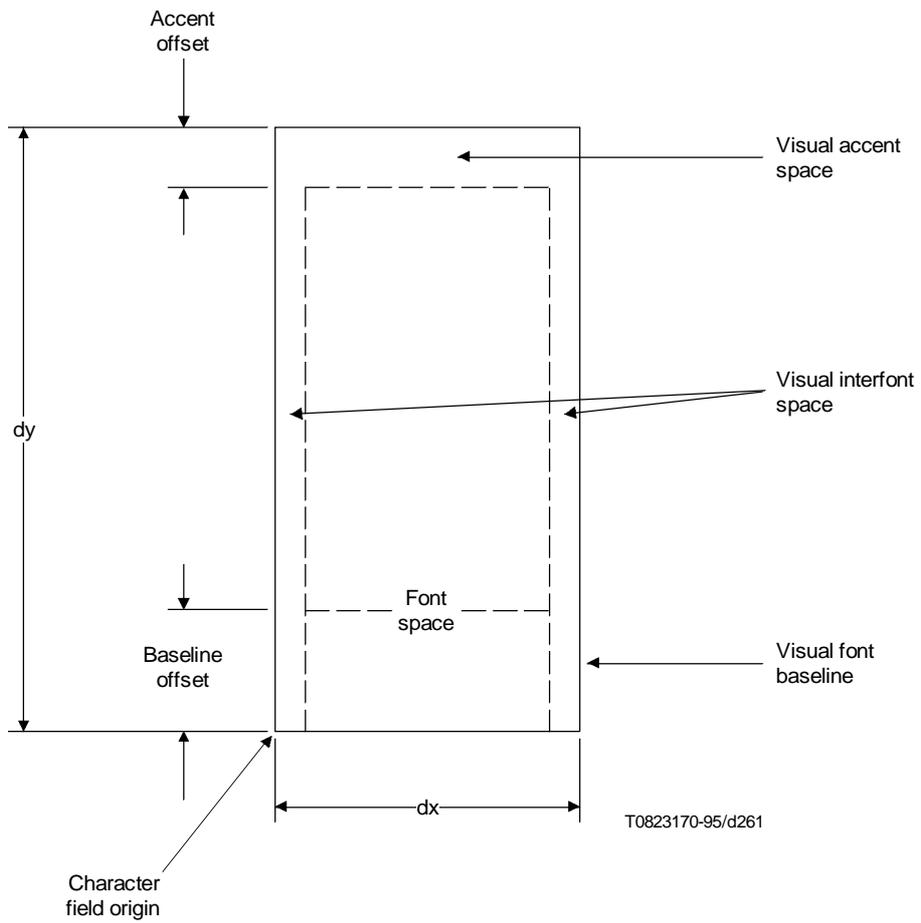
These Service Reference Models (SRM's) define the sets of specific implementation requirements for the videotex and teletext services. These sets of requirements represent the maximum functionalities that the information provider should assume when encoding text and pictorial information. These sets also represent the minimum functionalities that a receiving display device manufacturer should implement in order to ensure the satisfactory display of text and pictorial information. Equipment and information interchange meeting the requirements of these SRM's shall also meet the Conformance Requirements of this Data Syntax.

The SRM's are combined in Table IV.1. The requirements are identical in practically all cases. Where they differ, the particular parameters for the videotex and teletext services are specified separately. The minor differences arise primarily because of the one-way nature of the teletext service versus the two-way nature of the videotex service. Certain features, such as UNPROTECTED FIELD, are meaningful only in the videotex service and are therefore not used in the teletext service.

The requirements listed in Table IV.1 define the functionalities guaranteed (by a receiving display device that meets the requirements of the respective SRM) to the information provider and service operator for the generation and display of information. Normally, it is expected that the information provider would create information and the manufacturer would produce receiving display devices meeting these requirements. However, the following variants may arise:

- 1) The information provider may create information exceeding the respective SRM requirements. In such a case, a receiving display device meeting the SRM requirements may be expected to display the additional features in a reasonably satisfactory manner. However, such display is not always guaranteed. For example, the information provider may reasonably use a colour palette of 4096 colours if they are selected so that receiving display devices having a colour palette of only 512 colours give a satisfactory display.
- 2) A receiving display device may exceed the requirements of the respective SRM. Such a receiving display device may produce a more pleasing display which does not depend upon direct utilization of the additional requirements in the information created by the information provider. For example, a receiving display device that has a horizontal resolution of 512 pixels may produce more pleasing images than a receiving display device with a horizontal resolution of 256 pixels, even though the information provider is using an address range of only 256 pixels.
- 3) A receiving display device may implement features to a lesser extent than defined in the Service Reference Model and still conform to this Data Syntax just as a monochrome television set conforms to the National Television System Committee (NTSC) colour television standard. It is cautioned that in such a case some information may be lost just as a monochrome television receiver loses the colour 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 IV.1 have been identified as necessary to define an SRM for videotex and an SRM for one-way teletext. The conformance clause of this Data Syntax indicates which functions shall be executed and presented exactly and which can be presented in an approximate form.



**Figure IV.1/T.101 – Character field layout**

**Table IV.1/T.101 – General Videotex and Teletext Service Reference Models (SRMs)**

Functions	Requirements
1. Code extension sequences	All code extension sequences defined in this Data Syntax (consistent with the lower layers of the service) shall be executed [see D.8.3.3, item 1)].
2. Primary character set	<p>All 94 primary characters shall be uniquely presented for character field sizes greater than or equal to <math>dx = 6/256</math>, <math>dy = 8/256</math> [see D.8.3.3, item 2)].</p> <p>The offset of the visual font baseline, i.e. the apparent bottom of upper case alphabetic characters (see Figure IV.I) shall be 20% of the height (<math>dy</math>) of the current character field size, consistent with the physical resolution.</p> <p>The accent space (see Figure IV.I) shall be sufficient for a discernible, but not necessarily distinguishable, display.</p> <p>It is recommended that, if possible, the font be centered within the character field, leaving an equal amount of visual interfont space to the left and right of the font, but in any case, there shall be some such space to the right of the font.</p> <p>All other aspects of the character font are implementation-dependent.</p>
3. All remaining characters of the graphic character repertoire	<p>All remaining characters of the graphic character repertoire (see D.7.1) shall be presented.</p> <p>Character font shall be visually consistent with the primary character set but depends on implementation [see D.8.3.3, item 3)].</p>
4. PDI set, geometric drawing primitives	<p>All geometric drawing primitives shall be executed and presented as specified in D.8.3.3, item 4.</p> <p>The limit on the number of vertices in a polygon and in a spline shall be 256. (See D.5.3.3.3.1, D.5.3.3.5.1 and D.5.3.3.6.5).</p>
5. PDI set, attribute control functions	<p>All attribute control functions shall be executed as specified in D.8.3.3, items 4), 5) and 6), and presented as below:</p> <p>1) DOMAIN</p> <p>a) Single-value operand length</p> <p>b) Multi-value operand length</p> <p>c) Dimensionality</p> <p>d) Logical pel</p> <p>e) Drawing execution</p> <p>2) TEXT</p> <p>a) Character rotation</p> <p>b) Character path movement</p>
1) DOMAIN	All single-value operand lengths shall be executed.
a) Single-value operand length	All multi-value operand lengths shall be executed.
b) Multi-value operand length	Shall execute and present both the two and the three-dimensional modes. However, the third dimension shall be ignored when in the three-dimensional mode.
c) Dimensionality	The full range of logical pel heights and widths, consistent with the physical resolution, shall be executed and presented.
d) Logical pel	The position of the drawing point and cursor shall be maintained, consistent with the physical resolution (see item 10 of this table and Appendix II).
e) Drawing execution	All four orientations (i.e. 0 degrees, 90 degrees, 180 degrees and 270 degrees) shall be executed and presented.
2) TEXT	All four directions (i.e. left, right, up, and down) shall be executed and presented.
a) Character rotation	
b) Character path movement	

Functions	Requirements
c) Intercharacter spacing	All four spacings (i.e. 1, 5/4, 3/2, and proportional) shall be executed and presented, consistent with the physical resolution. The spacing of proportional spacing is implementation-dependent, as font is not defined.
d) Inter-row spacing	All four spacings (i.e. 1, 5/4, 3/2 and 2) shall be executed and presented, consistent with the physical resolution.
c) Move attributes	All four attribute combinations (i.e. move together, cursor leads, drawing point leads, and move independently) shall be executed.
f) Cursor style	All four styles (i.e. underscore, block, cross-hair, and custom) shall be executed and presented.
g) Character field sizes	All character field sizes greater than or equal to $dx = 6/256$ , $dy = 8/256$ (see Appendix II) shall be executed and presented consistent with the physical resolution, which includes the following character display formats (column $\times$ row): $40 \times 24$ , $40 \times 20$ , $40 \times 10$ , $32 \times 16$ , $20 \times 10$ .
3) TEXTURE	
a) Line texture	All four textures (i.e. solid, dotted, dashed, and dotted-dashed) shall be presented.
b) Texture pattern	All four defined masks (i.e. solid, vertical hatching, horizontal hatching, and cross-hatching) shall be presented.  The four programmable texture masks shall be presented up to a resolution of $16 \times 16$ stored elements.
c) Highlight	The highlight attribute shall be executed as specified in D.5.3.2.4.3 in all colour modes.
4) COLOUR	All colour modes (i.e. 0, 1, and 2) shall be presented. Sixteen simultaneous colours out of a set of 512 obtained by allocating three bits each to G R and B shall be available. The exact values of the default colours are not guaranteed due to the round of errors in the colour equation (see D.5.3.2.5.2).  The reference for the chrominance and luminance of the primary colours is that defined by the NTSC system (FCC Rules and Regulations, Part 73 Radio Broadcast Services, Subpart E).
5) BLINK	BLINK commands from the PDI set and the C1 set shall be executed and presented. Sixteen blink processes shall be available.
6) WAIT	<i>Videotex</i> : The duration of a minimum wait interval (operand of zero, see D.5.3.2.8.2) shall be between 0 and 1/10 of a second, inclusive.  <i>Teletext</i> : The duration of a minimum wait interval (operand of zero, see D.5.3.2.8.2) shall be between two complete TV fields (approximately 1/30 of a second) and three complete TV fields, inclusive.
7) RESET	This command shall be executed and presented as described in D.5.3.2.9.

Functions	Requirements
6. Mosaic set	All separated and contiguous mosaics shall be uniquely presented for character field sizes greater than or equal to $dx = 6/256$ , $dy = 8/256$ [see D.8.3.3, item 7)].
7. Macro set	<p>The number of macros definable shall be 96, subject to memory limitation. Macros may be accessed by user input mechanisms, for example, keys. Transmit-macros that are linked to user input are always available for user activation. Other macros that are linked to user input are only available for user activation when the cursor is on. Execution of a macro by user activation may result in an implementation-dependent state of the receive presentation process. A macro activated in this manner shall be executed in a contiguous sequence to completion. It is the responsibility of the service provider to restore the receiving device to a known state, if desired.</p> <p><i>Videotex:</i> A minimum of eight such linkages is required to activate macros 2/0 to 2/7. It is recommended that the service provider define non-linked macros starting at 7/15. A linkage may be represented, for example, by an individual key or a keying sequence. A transmit-macro shall be considered as a communication to a host computer [see D.8.3.3, item 8)].</p> <p><i>Teletext:</i> A transmit-macro shall be considered as a transmission to a local process [see D.8.3.3, item 8)].</p>
8. DRCS	The number of DRCS characters definable shall be 96, subject to memory limitation. The minimum resolution of the storage buffer shall be the minimum physical resolution (as specified in item 10 of this table) covered by the character field at the beginning of the DRCS definition. For example, if the character field size is $dx = 6/256$ and $dy = 10/256$ , then the storage buffer shall be an array of at least 6 elements horizontal by at least 10 elements vertical. This minimum resolution is assumed by the memory requirements in item 11 of this table. The TEXT attributes defined in item 5,2) of this table shall apply during definition and display of DRCS [see D.8.3.3, item 9)].
9. C0 CONTROL set	<p>The set shall be executed and presented as described in D.8.3.3, item 10).</p> <p><i>Videotex:</i> The lower layers of the service do not guarantee that the CAN character will be processed [see D.8.3.3, item 10)], unless the CAN immediately follows the macro invocation.</p> <p><i>Teletext:</i> The lower layers of the service do not guarantee that the CAN character will be processed [see D.8.3.3, item 10)].</p>
10. Physical display parameters	<p>Resolution shall be on the order of 256 pixels horizontal by 200 pixels vertical. The full X dimension (width) of the unit screen, and the Y dimension (height) from 0.0 to 0.78125, shall be visible in the display area.</p> <p>Implementation of the border area is not guaranteed (see D.4.2.2).</p>

Functions	Requirements
11. Memory requirements	<p>The receiving device shall provide at least 32 bytes of storage organized as 16 × 16 pixels for each programmable texture mask. This storage is not included in the totals described below.</p> <p><i>Videotex:</i> The total storage of macro definitions, DRCS characters, and unprotected fields shall not require more than 3 kilobytes (3072 bytes). This total does not include implementation overhead in the receiving device, and assumes the following:</p> <ul style="list-style-type: none"> <li>a) The storage of macro definitions requires one byte for each byte of defining code excluding the DEF and terminating codes.</li> <li>b) Three bytes of storage are required for each complete character cell in unprotected fields.</li> <li>c) The storage of DRCS is such as to allow 96 characters of NORMAL size to be defined in the shared memory and still leave 2 kilobytes (2048 bytes) of that memory for use by macro definitions and unprotected fields, provided that the total amount of presentation layer code defining the DRCS characters does not exceed 3 kilobytes (3072 bytes).</li> </ul> <p><i>Teletext:</i> The total storage of macro definitions and DRCS characters shall not require more than 3 kilobytes (3072 bytes). This total does not include implementation overhead in the receiving device. Assumptions similar to those made in the videotex SRM are made, but are more fully specified for this and other memory limitations in NABTS.</p>
12. C1 CONTROL set	<p>Executes and presents the following: NORMAL TEXT, MEDIUM TEXT, DOUBLE HEIGHT, DOUBLE SIZE [see D.8.3.3, item 11)]. Characters following a SMALL TEXT control (4/10) shall be executed but may not be presented. Buffering of data scrolled out of the display area or active field shall not be required [see D.8.3.3, item 11)].</p> <p><i>Videotex:</i> Executes all control codes from the C1 code set as defined in this Recommendation.</p> <p><i>Teletext:</i> All control codes from the C1 set shall be executed except for the following whose execution is implementation-dependent:  5/0 PROTECT  5/15 UNPROTECT</p>
1) Unprotected fields	<p><i>Teletext:</i> Not applicable. The execution of the PROTECT and UNPROTECT commands shall have no effect.</p> <p><i>Videotex:</i> The number of unprotected fields available shall be 40, subject to memory limitation.</p> <p>Only characters from the graphic character repertoire, from the mosaic set (including separated mosaics), and from the DRCS set may be stored for the purpose of editing and subsequent transmission (see Note 1). Any such characters are required to have a uniform size that is the same as the text size established when the field was unprotected, as well as the default text attributes for rotation, character path, intercharacter spacing, inter-row spacing, and reflection in order to be guaranteed to be stored. This uniform character size defines a grid which is aligned with the field origin. Characters are guaranteed to be stored only when their respective character fields align with this grid (see Note 2). The storage and display when such alignment does not occur is implementation-dependent, e.g. the character may be moved into alignment and stored, or the character might not be moved or stored. In addition, it is only guaranteed that characters received from the host will be stored if the active field coincides with the unprotected field being written into by the host. Characters which fall in an unprotected field but do not satisfy the above condition concerning attributes may be handled in an implementation-dependent manner.</p> <p>All other presentation level data are displayed according to the normal presentation display process but is not stored.</p>

Functions	Requirements
	<p>Colour is permitted to be different for each character field within an unprotected field.</p> <p>Colour mode is permitted to be different for each character field within an unprotected field.</p> <p>The colours stored for the colour attribute in unprotected fields are the colours as they appear in the colour map and appear to the user. The colour map itself is only transmitted back to the host indirectly if and when colour mode 0 is used for transmission.</p> <p>DRCS pattern definitions need not be stored with an unprotected field or subsequently transmitted. Such definitions may not be interpreted if transmitted to a host. Only those unprotected fields which have been edited by user (keyboard) interaction need be transmitted (see Note 3).</p> <p>Data can be entered and/or edited in unprotected fields by user interaction only if the display cursor is on. User entry of data into unprotected fields shall not interfere with the reception and proper interpretation of received presentation layer code. It is permissible for the transmission of the data in unprotected fields (initiated by the user) to turn the display cursor off. The contents of the unprotected field should be maintained until the unprotected field is protected. When no unprotected fields are defined, or when the display cursor is off, an implementation-dependent communications mode exists.</p> <p>The transmission format is shown in Table IV.2 (see Notes 1 and 4).</p>
<p><b>NOTES</b></p> <p>1 The transmission of data from the receiving display device to the host is a separate presentation level process independent of the host to receiving display device presentation process. Table IV.2 defines the presentation layer code that the terminal may use to transmit the data stored in the unprotected field and the maximum which a host is required to interpret. If additional codes are transmitted that exceed this SRM, a host may ignore them. A terminal should not send data in such a way that a host that interprets only up to the SRM limit will become confused by any additional data.</p> <p>2 The alignment of the size and position of characters within an unprotected field cannot be guaranteed if different multivalued operand lengths are used to specify the field and the character. If the size specified is NORMAL, SMALL, DOUBLE HEIGHT, or DOUBLE SIZE, alignment of positioning is only guaranteed if Format Effectors are used to position characters relative to the origin of the field.</p> <p>3 When the user initiates transmission of an unprotected field, at least the information required for every transmission should be sent (see Table IV.2), even if no unprotected fields are sent.</p> <p>4 Table IV.2 indicates the transmission format of the unprotected field information. As indicated in the condition column, some of the codes are optional and are only required when necessary to ensure that the code conforms to this Data Syntax and accurately represents the image stored in the unprotected field (within the constraints previously discussed). Different applications may have different input requirements. It is desirable for terminals to attempt to minimize the amount of transmitted code, and that each field be context-independent of other fields. In particular, re-establishment of the default designations and invocations of G-sets in each field is important for context independence.</p> <p>5 The coordinate specification of the origin of the field must be the same for transmission as that which was used to define the field.</p>	

**Table IV.2/T.101 – Unprotected field transmission format**

Occurrence	Code	Condition (see Note 4 of Table IV.1)
Per unprotected field block transmission	NSR	Required at the beginning of a transmission. Optional before other fields.
	DOMAIN	Required only if different from the default.
Per unprotected field	FIELD (see Note 5 of Table IV.1)	Required for each field.
	TEXTE	Required only if text size is different from the default or that specified in the last field.
	DÉTERMINATION POINT (ou caractère de commande de mise en page)	Required only to locate the first character within the field if not already located.
Par caractère dans un champ sans protection	SÉLECTION COULEUR (ou DÉTERMINATION COULEUR en mode de couleur 0)	Required only if colour changes.
	Codes de caractère (y compris les séquences d'extension de code appropriées) et tout caractère de commande de mise en page, ESPACE, RÉPÉTITION, RÉPÉTITION JUSQU'À EOL	Required only to position characters within the field.
	FIN	Required at the end of each field.
	DÉTERMINATION POINT	Required to establish, for the host, the position where the next user input would occur.
	SDC	Required to logically delimit the end of the unprotected field transmission.

**Addendum**  
**to**  
**Data Syntax III for international interactive videotex service**

*(1993)*

## **Preface**

Since the approval and publication of the CCITT Recommendation T.101, Data Syntax III (1985), which corresponds to the Standard: Videotex/Teletext Presentation Level Protocol Syntax (NAPLPS), a number of interpretation and clarification issues have been dealt with. These interpretations and clarifications, together with up-dating of the Code Extension due to revision of the ISO 2022 Standard, are now embodied in this addendum.

## **1 Underline**

With reference to D.6.2.7.15 and D.5.3.2.3.9 of Recommendation T.101, Annex D, Data Syntax III, the underline mode is added to the character before any reflection or inversion. It is therefore placed at the bottom of the character, no matter what the position or orientation of the character cell.

Since underline mode is applied to the character, it cannot be drawn outside the character field. The thickness of the underline is the absolute value of the *dy* of the logical pel – but it cannot exceed the *dy* of the character field. The underline starts at the origin as stated in Recommendation T.101, Annex D, Data Syntax III. If the *dy* of the logical pel is zero, an underline of one pixel width is drawn.

## **2 Absence of operands on PDIs**

For some PDIs and controls, Recommendation T.101, Annex D, Data Syntax III clearly states the action to be taken when operands are missing. Consistent with D.5.3.2.2.5, Annex D, Data Syntax III, for all PDIs, when not otherwise stated, non-existent operands are assumed to be zero.

## **3 RESET PDI, NON-SELECTIVE RESET (NSR) AND ACTIVE POSITION SET (APS)**

As indicated in the section D.5.3.2.9.3 and Table D.28 of Recommendation T.101, Annex D, Data Syntax III, a PDI RESET where bit 1 of byte 2 is set to 1 causes the active field to become the unit screen. Similarly, as stated in D.6.1.6.5, NSR causes the active field to become the unit screen.

If the parameters of the NSR command indicate that the cursor is to be positioned outside the unit screen, it is constrained to the unit screen consistent with D.5.3.2.3.7, Annex D, Data Syntax III. Similarly if part of the character cell is placed outside of the unit screen by the use of a double height or other attribute, repositioning consistent with D.5.3.2.3.7 applies.

The effective grid generated by the NSR positioning parameters relative to the position of graphics, or to the position of characters defined using the APS command, may not always register the same since the NSR grid depends on the display area. The APS command may position the cursor into the unit screen outside of the display area.

## **4 Cursors**

Recommendation T.101, Annex D, Data Syntax III requires that the cursor display be visible when the cursor is on, however the colour of the cursor is implementation dependent and the visibility of the display cursor is not guaranteed in all situations.

## 5 Macro linkages

Macros that are linked to user input are always available for user activation (refer to D.6.2.7.19 and Table IV.1, item 7). Control of user input is not at the presentation layer.

## 6 Logical pel dimensions

As documented in Recommendation T.101, Annex D, Data Syntax III, when the logical pel size is (0,0) for a textured line or a textured pattern, the texture defaults to solid, that is, no texture.

Also as documented in Recommendation T.101, Annex D, Data Syntax III, when the logical pel size is (0,0), the physical rendition of drawing functions such as LINE have a representation of one pixel.

When the logical pel size is (0,0), the separation of separated mosaics is the minimal physical separation which may be presented; that is, a separation of one physical pixel.

For a logical pel size of (N,0) or (0,N), that is, when only one dimension is zero, the minimum rendition occurs in the indicated direction consistent with the above. Texture patterns are solid in the direction of 0 logical pel, line widths are the minimum one pixel width, and the separation for mosaics is the minimum one pixel separation.

## 7 Word wrap

When the line length is shorter than a word and when the word contains more than one special terminating character, the terminating characters should be tested for first, since this produces consistent displays. DRCS characters are part of a word and not a terminator. SPACE is a terminator character for word wrap.

## 8 INCREMENTAL commands

The colour map is affected by the INCREMENTAL POINT command in a consistent manner with the effects of other commands. The colour map or colour map address operate exactly as though a SET COLOUR (in Colour Mode 0) or a SELECT COLOUR (in Colour Mode 1 or 2) was processed.

The limit of the number of coordinates for an INCREMENTAL POLYGON command is the same as for the POLYGON command, and is 256. The final image of an INCREMENTAL POLYGON is identical to the final image of an equivalent POLYGON command.

## 9 Unprotected fields

The issues relating to the creation, editing and retransmission of unprotected fields are very complex and are left for further study.

Further development of unprotected fields should be addressed by the application layer rather than by the presentation layer. Unprotected fields issues are implementation dependent.

## 10 Cursor and macro

In the attempt to develop Recommendation T.101, Annex D, Data Syntax III as a fully workable presentation protocol in the absence of other layers, a number of functions have been assigned to presentation entities that ideally belong to other layers. In particular, the "cursor" performs a dual function of:

- i) providing a visual feedback for user input; and
- ii) controlling the user linked display macro keys.

Recent development of service specific application layers have made it possible to move certain functions from the Data Syntax to the application layer.

In the case in which Recommendation T.101, Annex D, Data Syntax III operates along with an application layer protocol that provides its server with control of user input, the cursor function with Data Syntax III may be unlinked from control of display macros.

## 11 Proportional spacing of text

The exact amount of space that each character occupies when using proportional spacing is not standardized as part of the Recommendation T.101, Annex D, Data Syntax III. However, page creators encountered great difficulty because they were not able to predict the length of a line of proportionally spaced text on terminals of different manufacture. In order to solve this problem, the following guideline was developed which defines the width of each proportionally spaced character in terms of a number of categories. This guideline for the effective use of proportionally spaced text should be followed in the composition and display of pages in which wrap or scroll would cause undesirable results.

To calculate the maximum length of any string of proportionally spaced text, a function is herein defined to determine the distance the cursor is moved if the character field width is parallel to the character path. This function applies only to the character repertoire as defined in clause D.7 of Recommendation T.101, Annex D, Data Syntax III.

NOTE – The cursor is moved equal to the character field height and path are parallel.

This calculation is based on an algorithm which categorizes the characters of the repertoire into ten distinct classes. Every time a character is displayed, the displacement of the cursor is calculated by applying the function to the width class of the character and to the current character field width.

This displacement allows sufficient room for both the character font space and the visual interfont space.

### 11.1 Width tables

Tables 1 and 2 define the class for each of the characters in the Primary Character Set and the Supplementary Character Set. The class for the SPACE character is defined to be 9 (full width). The width class of a composite character created by the use of non-spacing character together with a spacing character, is determined by the maximum width class of its components. For example, the accented character  $\hat{i}$  (coded as SS2, 4/3, 6/9) would have the width class of the circumflex (^), that is class 2. Note that a non-spacing accent combined with SPACE would have the width class 9 (full width). The combination of underline (S 4/12), vector overbar (S 4/0) or non-spacing slant (S 4/9) with any character will also have a width class of 9. The format effector characters APB and APF in proportional space mode are considered to have a width class of 9 (full width).

In an analogous manner to the positioning of text and graphics (see Appendix II of Recommendation T.101, Annex D, Data Syntax III), the accumulated displacement of a string of proportionally spaced text should be maintained to within one pixel accuracy, consistent with the physical resolution. This means that the positioning accuracy for each individual character must not result in accumulated error.

Because of variations that may occur in implementation, it is good practice to avoid active fields which tightly bracket proportionally spaced text. Also note that since this algorithm only bounds the length of a string of proportional spaced text, the final cursor position is implementation dependent.

### 11.2 The displacement function

The function which calculates the displacement value is based on two input parameters: the width class of the character being displayed, and the dx value of the current text field size. The result produced by the function is an always-positive value which represents the magnitude of the displacement which is then used to move the cursor in the appropriate direction.

The ten width classes are used to create either five or seven unique character widths and cursor displacements, depending on the character size.

**Table 1/T.101 – Proportionally spaced primary character widths**

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

**Table 2/T.101 – Proportionally spaced supplementary character widths**

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

### 11.3 Features of the function

In calculating the displacement, the function maintains the following goals:

- a) For a given text size, the displacement for characters of the widest character class equals the current character field dx, truncated to the current text size's spacing unit dimension.
- b) The function permits the interfont gap between all characters in a text string to be identical, in order to insure good string appearance.
- c) The function guarantees that normal size text on a low resolution terminal will appear identical in spacing to the same text displayed on a higher resolution terminal.
- d) On terminals which have a resolution equal to an integer power of 2, all font displacements will be in integer power of 2, all font displacements will be in integer pixel distances, thus maintaining good string appearance.

### 11.4 The function

The displacement calculation is divided into two ranges of text sizes. The first range is  $0 \leq dx < 12/256$ , while the other is  $12/256 \leq dx < 1$ . The first range encompasses those text sizes which may not be displayable on terminals of all resolutions. The function handles these sizes in a way that guarantees the third feature listed above. The second range encompasses sizes which are likely to be displayable on all available resolutions, and provides a more continuous range of displayable text sizes.

### 11.5 Text sizes $0 \leq DX < 12/256$

The function addresses all text sizes in the range  $0 \leq dx < 12/256$ , by scaling a fixed set of six predefined font spacing tables.

**Font spacing tables for  $0 \leq dx < 12/256$   
in spacing units**

Font size	Width class									
	0	1	2	3	4	5	6	7	8	9
6	2	3	4	3	4	5	6	4	5	6
7	3	4	5	4	5	6	7	5	6	7
8	2	3	4	4	5	6	7	6	7	8
9	3	4	5	5	6	7	8	7	8	9
10	4	5	6	6	7	8	9	8	9	10
11	3	4	6	6	7	8	0	8	0	11

Each font spacing table defines the cursor displacement, in terms of base spacing units, to be applied to characters in each of the ten width classes. The six tables cover a range of text sizes from 6 units wide, up to but not including 12 units wide, giving the tables an overall dynamic range of one binary order of magnitude. To determine the displacements for a given text size, the base spacing unit values in one of the tables is multiplied by a scaling factor, which is always some power of two.

Given a text size whose  $dx$  is less than  $12/256$ , the proper scaling table is selected by first finding the scale factor, which must be some power of two, such that the text  $dx$ , in unit coordinates, divided by the scale factor is found, the proper table will be indicated by truncating the fractional portion of the value for  $dx/\text{scale factor}$ . The resultant integer will be between 6 and 11. This integer corresponds to the font size indicated by the above table.

Once the proper font spacing table has been selected and the scale factor has been determined, the unit coordinate displacement for each character class is determined by simply multiplying the indicated unit spacing in the table for that width class by the scale factor.

**11.6 Text sizes  $DX \geq 12/256$**

Text sizes of  $12/256$  and larger are handled by scaling a second font spacing table, using a derivative of the text  $dx$  as a scaling factor, and using the result as an adjustment to the current text size  $dx$ .

**Font spacing table for  $dx \geq 12/256$  in spacing units**

Width class									
0	1	2	3	4	5	6	7	8	9
6	5	4	4	3	2	1	2	1	0

The scaling and the resultant displacement are such that font patterns can always remain symmetric (on characters such as M and W) by always scaling to odd pixel widths, and also maintain identical interfont gaps between all characters of a given size.

The first step to calculate a displacement is to determine the scale factor used to scale the font spacing table. The scale factor is determined by the following:

- 1) Truncate the unit coordinate text size  $dx$  to the equivalent of DOMAIN 3, leaving the eight most significant bits of the unit coordinate. Call the result N.
- 2) Multiply N by the factor 11/13. Be careful to avoid register overflow.
- 3) Subtract the equivalent of  $1/256$  from the result, bitwise or the result with  $1/256$ , then again subtract  $1/256$  from the result. Truncate the result to the equivalent of DOMAIN 3 (8 bits). Call the result the scale factor F. This step has the effect of permitting the font patterns for the widest character class in a given text size to always be scaled to an odd width.

Next, the unit coordinate displacement is calculated as follows:

- 1) Find the unit spacing for the proper width class in the font spacing table for  $dx \geq 12/256$ .
- 2) Multiply the unit spacing by the scale factor F.
- 3) Divide the result by 6, add the equivalent of  $1/512$  to effect half-rounding, then truncate the result of the equivalent of DOMAIN 3, leaving the eight most significant bits of the unit coordinate. Finally, subtract this value from N. This result is the desired cursor displacement in unit coordinates.

## 12 Character field sizes

The following “Note” is to be added to Table IV.1 (SRM), item 5, sub item 2, g) of Recommendation T.101, Data Syntax III:

NOTE – Field sizes greater than  $6/256$  by  $8/256$  also include display format  $42 \times 25$  (column  $\times$  row). The character field size that displays as  $40 \times 24$  requires a higher than default domain to define.

## 13 Code extension

The following are not issues of clarification but are the results of the revision of the ISO 2022 Standard and registrations of the Complete Code and the Control Sets and G-Sets carried out since the publication of Recommendation T.101, Annex D, Data Syntax III (1985), and as a result of the revision of Recommendation T.50 and the creation of Recommendation T.51. All new implementations should take into account these items, but current implementations need not make any changes.

### 13.1 Subclause D.4.3.1.1

Subclause D.4.3.1.1 of Recommendation T.101, Annex D, Data Syntax III, is replaced by the following revised text:

**D.4.3.1.1** The method of code extension used in this Data Syntax is based on the code extension techniques specified in ISO 2022-1986.

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- and C-sets to be switched.

Using the above code extension technique, any control or graphic set that is registered in accordance with ISO 2375 can be designated and invoked and used in conjunction with any or all the control and graphic sets (including the PDI set) of this Data Syntax, within the context of the code environment defined by the standard. Specifically, using this technique, this Data Syntax can cater for other linguistic requirements such as Greek, Cyrillic, Hebrew, Chinese Hanzi, Japanese Kanji and Katakana, Arabic, etc. The primary Latin alphabet used in this Data Syntax (see D.5.1) is that of the IRV of defined in Recommendation T.50 and the supplementary character set used (see D.5.2) is that defined in Recommendation T.51. The character sets identified in Recommendation T.52 may also be used in accordance with the rules of code extension defined in ISO 2022.

### 13.2 Subclause D.4.3.1.2

Subclause D.4.3.1.2 of Recommendation T.101, Annex D, Data Syntax III, is replaced by the following revised text:

**D.4.3.1.2** The entire coding environment described in this Data Syntax is to be designated and invoked as a “complete code” by the escape sequence ESC 2/5 4/1, in accordance with ISO 2022. Conforming interchange does not require the use of this escape sequence except when interchanging with other services. In this complete code, special attention has been paid to the following:

- G0: A 94 code position G-set
- G1: A94 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 positions 2/0 and 7/15. When such a set is invoked into columns 2 to 7, these two positions shall have the meanings of SPACE and DELETE, respectively.

A 96 code position code 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 an ESC 2/5 4/0 sequence or by the designation and invocation of any other “complete code” by an ESC 2/5 F sequence.

### 13.3 Subclause D.4.3.2

Subclause D.4.3.2 paragraph 5 of Recommendation T.101, Annex D, Data Syntax III, following Table D.1, is replaced by the following revised text:

The designation and invocation sequences for C0 and C1 sets are ESC 2/1 4/11 and ESC 2/2 4/6 respectively. All designation sequences designating G- and C1-sets defined in this Data Syntax are to be implemented. Other G- and C1-sets defined according to the rules of ISO 2022 may be implemented. All other designation sequences shall designate either a null G- or a null C1-set. The redesignation of the C0 sets is not permitted in the context of this Data Syntax and such redesignating escape sequences shall be ignored. A null set is a set in which all code positions are executed as null operations.

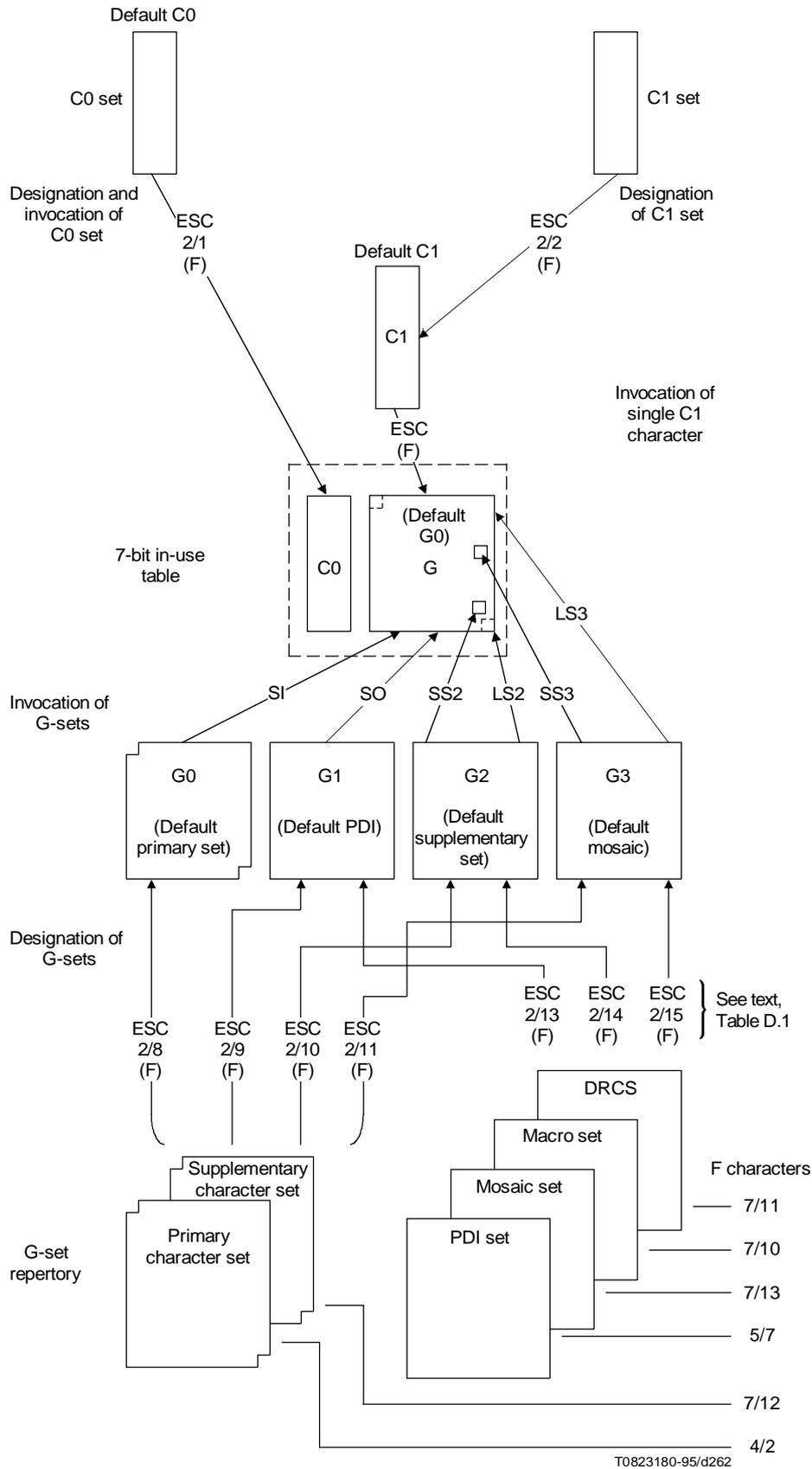
Subclause D.4.3.2 of Recommendation T.101, Annex D, Data Syntax III, Table D.1 and its Notes are replaced by the following:

**Table D.1/T.101 – Escape sequences for designation of C- and G-sets**

Escape sequence	Set to be designated
Control sets: ESC 2/1 4/11 ESC 2/2 4/6	C0 set C1 set
94-character sets: ESC I 4/2 ESC I 7/12  where I is 2/8, 2/9, 2/10, 2/11 for G0, G1, G2, G3, respectively	Primary character set 94-character Supplementary character set
96-character sets: ESC I 5/7 ESC I 7/13 ESC I 5/2  ESC I 2/0 7/10 ESC I 2/0 7/11  where I is 2/13, 2/14, 2/15 for G1, G2, G3, respectively	PDI set Mosaic set 96-character Supplementary character set Macro set <sup>a)</sup> DRCS set <sup>a)</sup>
<sup>a)</sup> The 4-byte escape sequence is in accordance with ISO 2022-1986 for DRCS. The Macro set is treated like the DRCS set. In accordance with ISO 2022-1986, only one set of I characters is used for designation sequences for character sets. However, to provide backward compatibility to ISO 2022-1982, terminal equipment manufacturers should continue to interpret the dual coding where I may also be 2/9, 2/10 and 2/11 corresponding to designations of G1, G2 and G3 respectively. Further, for the Macro set and DRCS set, the 3-byte sequences ESC I 7/10 and ESC I 7/11 respectively should also be implemented in terminals, in addition to the 4-byte sequences.	

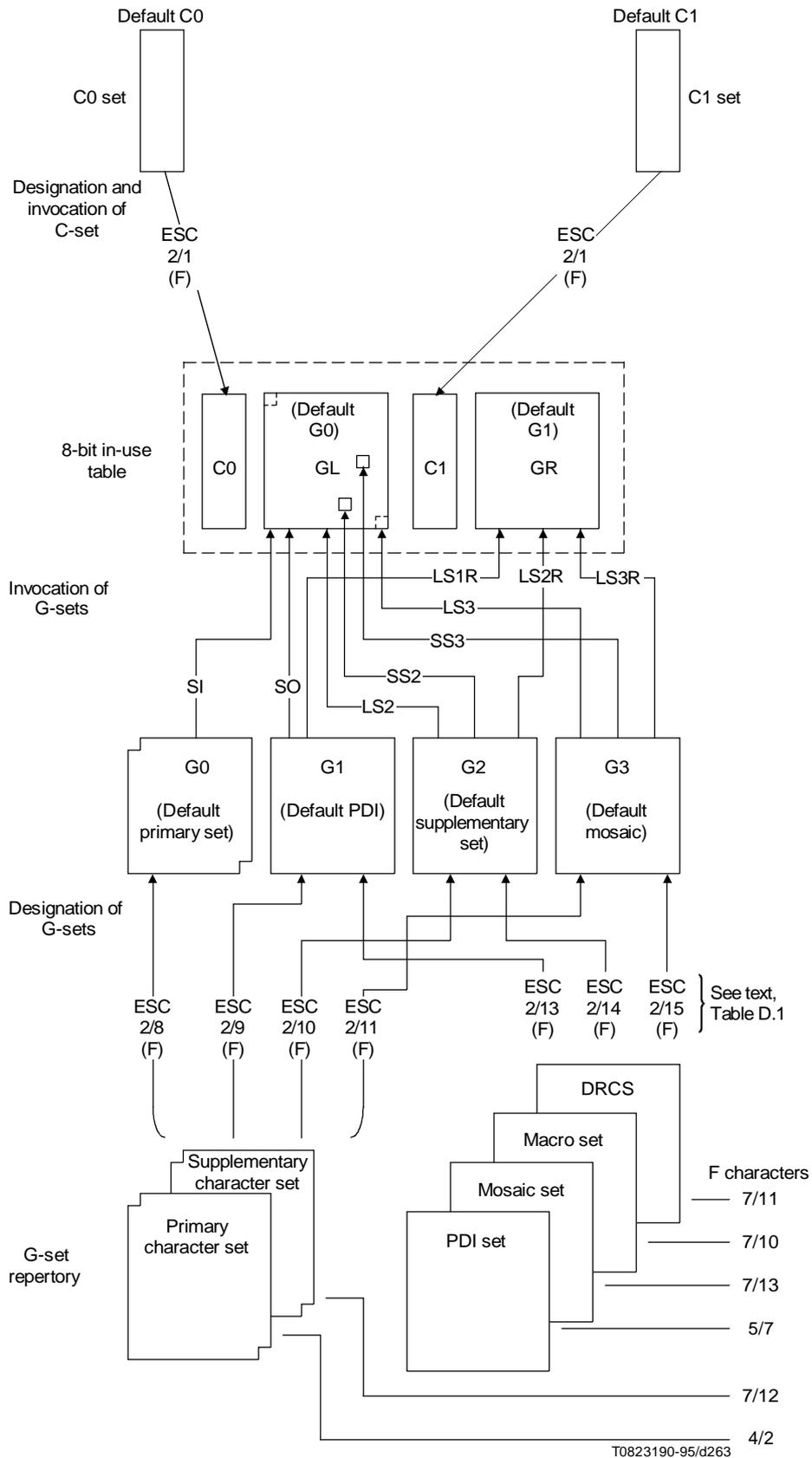
### 13.4 Figures D.4 and D.6

Figures D.4 and D.6 are replaced by the following revised diagrams:



NOTE – The 96-character supplementary set per Recommendation T.51 has the final character 5/2, also any of the character sets identified in Recommendation T.52 may be used in this coding environment.

Figure D.4/T.101 – Code extension in a 7-bit environment



NOTE – The 96-character supplementary set per Recommendation T.51 has the final character 5/2, also any of the character sets identified in Recommendation T.52 may be used in this coding environment.

Figure D.6/T.101 – Code extension in a 8-bit environment

## 14 ISO/IEC 9281 support

The following clause is added following clause D.9.

### D.10 ISO/IEC 9281 support

This Data Syntax supports the compatible extension to all ITU-T defined videotex data syntaxes through the picture coding environment mechanism specified in this Recommendation. All of the common extensions to the various ITU-T defined videotex data syntaxes are defined as picture coding environments per ISO/IEC 9281. Picture coding environments permit the inclusion of transparent data as is required by the common enhanced videotex photographic and audio data syntaxes defined by ITU-T.

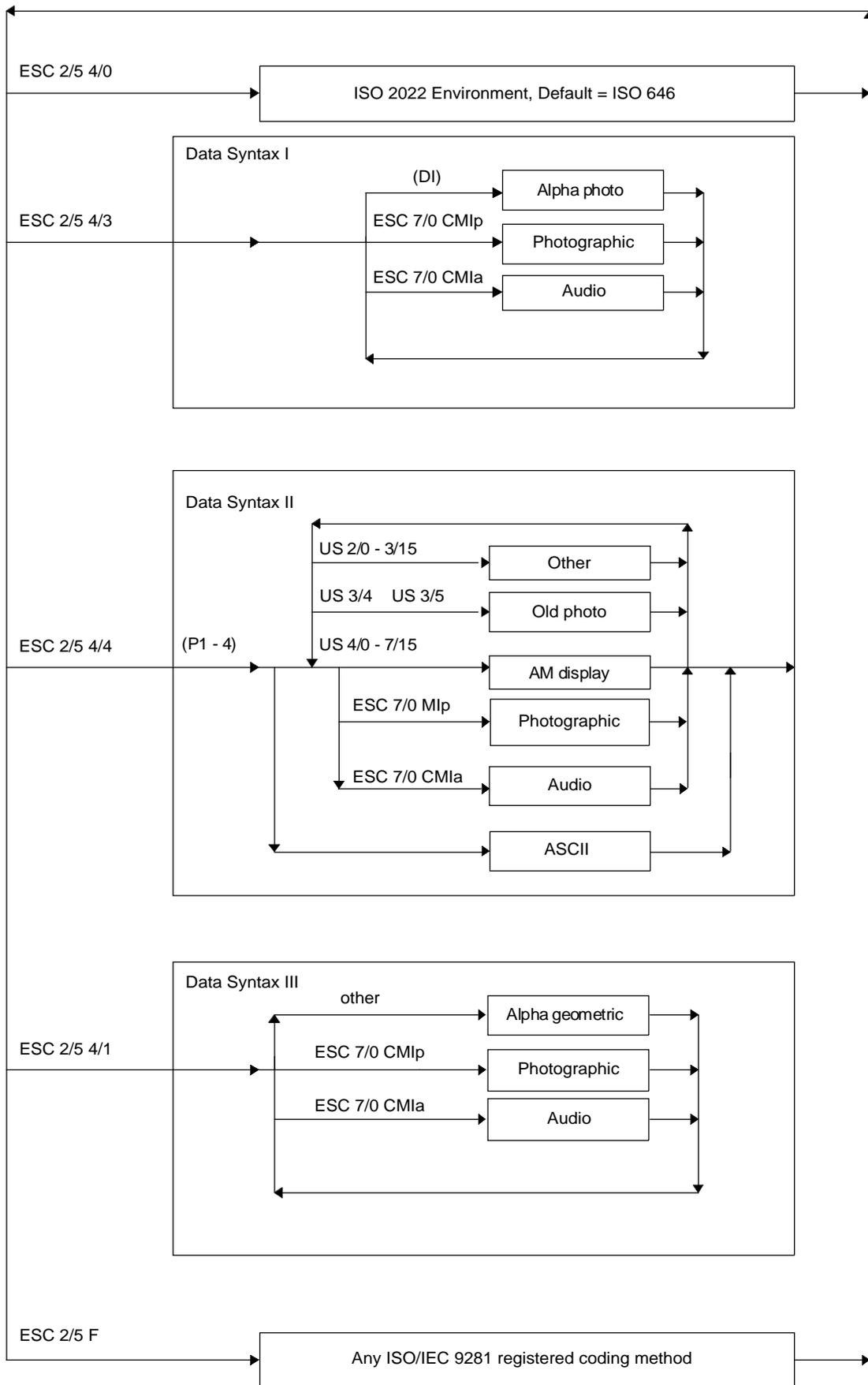
The ISO/IEC 9281 standard works compatibly with the concept of the complete coding environment specified in ISO 2022. Within a complete coding environment which supports ISO 9281 Picture Entities may be communicated.

A Picture Entity (PE), consisting of a Picture Coding Delimiter (PCD), a Coding Method Identifier (CMI), an optional Length Indicator (LI), and Picture Data Entity (PDE) may be followed directly by another PE or videotex data in the data syntax. The PCD is coded by the escape sequence ESC 7/0. CMI codes are registered by ISO. The LI comprises one or more bytes encoding the length of a data transparent PDE. At the end of a PE if another PE were not established, then one reverts back to the base videotex data syntax. This is illustrated by Figure D.67.

The ISO 9281 mechanism is used to support the ITU-T common videotex enhancements. Terminals which do not support these enhancements should support the structure of the extension mechanism and ignore any data following ESC 7/0 up to the end of the PDE as specified by the LI.

	Basic Videotex	Picture Entity – e.g. Audio				Basic Videotex
ESC 2/5 F	Videotex data	PCD ESC 7/0	CMI	LI	PDE Audio data	Videotex data

**Figure D.67/T.101 – Operation of ISO 9281 extension mechanism**



T0823200-95/d264

Figure D.68/T.101 – Global switching mechanism

## 15 Alignment with Recommendation T.51

The following are not issues of clarification but are the results of alignment with Recommendation T.51. All new implementations of the Recommendation should take into account these items, but current implementations need not make any changes.

### 15.1 Subclause D.5.2

Subclause D.5.2, paragraph 1, of Recommendation T.101, Annex D, Data Syntax III, is replaced by the following revised text:

#### D.5.2 Supplementary Character Set

The supplementary character set of accents, diacritical marks, and special characters for Latin-based alphabets is illustrated in Figure D.8, Part 1. This table is equivalent to the supplementary code table defined in ITU-T Recommendation T.51. The particular patterns (font) chosen for the characters are implementation-dependent and are constrained only by the specified character field at each character 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 D.5.3.2.3.4.

Subclause D.5.2, paragraph 3, of Recommendation T.101, Annex D, Data Syntax III, is replaced by the following revised text:

The supplementary character set illustrated in Figure D.8, Part 1, is a 96-character set in complete alignment with Recommendation T.51 and ISO Standard 6937. The implementation of this set is optional, but it is recommended for compatibility with other ITU-T defined Telematic services. The sequence used to designate this supplementary set is ESC I 5/2, where I is 2/13, 2/14 or 2/15 to indicate G1, G2 or G3 respectively (see D.4.3).

The supplementary character set illustrated in Figure D.8, Part 2, is a 94-character set derived from a previous version of this Data Syntax. It is retained for compatibility, and its implementation is mandatory. Certain little used characters which do not appear in Recommendation T.51 or ISO 6937 have been identified as being retained only for compatibility with older implementations. The support of these characters is optional. Two additional characters have been added to complete alignment with the basic 94 subset of Recommendation T.51. These characters are identified by notes along with the character set.

15.2 Figure D.8

Figure D.8 is replaced by the following two-part figure:

					10	11	12	13	14	15	
					b <sub>7</sub>	0	0	1	1	1	1
					b <sub>6</sub>	1	1	0	0	1	1
					b <sub>5</sub>	0	1	0	1	0	1
b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	Row	Col	2	3	4	5	6	7
0	0	0	0	00	NBSP	0	②	—	Ω	κ	
0	0	0	1	01	ı	±	˘	ˆ	Æ	æ	
0	0	1	0	02	ç	²	˙	®	Ð	đ	
0	0	1	1	03	£	³	^	©	á	ð	
0	1	0	0	04	②	x	~	™	Ĥ	ĥ	
0	1	0	1	05	¥	μ	—	♪	②	ı	
0	1	1	0	06	②	¶	˘	¬	ıı	ıı	
0	1	1	1	07	§	.	·	ı	Ł	ł	
1	0	0	0	08	¼	÷	..	②	Ł	ł	
1	0	0	1	09	‘	’	②	②	Ø	ø	
1	0	1	0	10	“	”	°	②	Œ	œ	
1	0	1	1	11	«	»	˘	②	Œ	œ	
1	1	0	0	12	←	¼	②	⅛	þ	Þ	
1	1	0	1	13	↑	½	”	⅜	ƒ	ƒ	
1	1	1	0	14	→	¾	˘	⅝	ŋ	ŋ	
1	1	1	1	15	↓	ı	˘	⅞	ˆn	SHY	

T0823210-95/d265

NOTES

- 1 Column 4(12) is non-spacing.
- 2 These character table positions shaded to indicate that they are empty and as such they should have the same presentation effect as SPACE. New characters may be assigned if Recommendation T.51 and ISO 6937 are revised.

Figure D.8/T.101 (Part 1) – 96-character supplementary character set

					10	11	12	13	14	15	
					b <sub>7</sub>	0	0	1	1	1	1
					b <sub>6</sub>	1	1	0	0	1	1
					b <sub>5</sub>	0	1	0	1	0	1
b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	Row	Col- umn	2	3	4	5	6	7
0	0	0	0	00		0	②	—	Ω	κ	
0	0	0	1	01	i	±	˘	ˆ	Æ	æ	
0	0	1	0	02	ç	²	,	®	Ð	đ	
0	0	1	1	03	£	³	^	©	à	ð	
0	1	0	0	04	② \$	X	~	™	ℍ	ℎ	
0	1	0	1	05	¥	μ	—	♪	②	ı	
0	1	1	0	06	② #	¶	˘	¬	ıĲ	ıĵ	
0	1	1	1	07	§	.	·	ı	Ł	ł	
1	0	0	0	08	¤	÷	¨	②	Ł	ł	
1	0	0	1	09	‘	’	② /	②	Ø	ø	
1	0	1	0	10	“	”	◦	②	Œ	œ	
1	0	1	1	11	«	»	◄	②	◌̇	β	
1	1	0	0	12	←	¼	②	⅛	ƒ	ƒ	
1	1	0	1	13	↑	½	"	⅜	ƒ	ƒ	
1	1	1	0	14	→	¾	˘	⅝	ŋ	ŋ	
1	1	1	1	15	↓	ı	˘	⅞	ˆn		

① | T0823220-95/d266

NOTES  
 1 Column 4(12) is non-spacing.  
 2 These character table positions are shaded to indicate that they are empty and as such they should have the same presentation effect as SPACE. New characters may be assigned if Recommendation T.51 and ISO 6937 are revised. The characters illustrated in these positions are retained for backward compatibility with older implementations of the Recommendation and their implementation is optional. The rectangles surrounding characters in 4/12, 5/8 through 5/11, and 6/5 are for illustrative purposes only and are not part of the graphic symbol.

Figure D.8/T.101 (Part 2) – 94-character supplementary character set

### 15.3 Repertoire

Subclause D.7.3, Table D.25 of Recommendation T.101, Annex D, Data Syntax III, is replaced by the following revised text:

**Table D.25/T.101 – 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
®	Registered sign	S 5/2
©	Copyright sign	S 5/3
™	Trade mark sign	S 5/4
♫	Musical note	S 5/5
`	Grave accent	6/10
^	Circumflex	5/14
~	Tilde	7/14
¬	Logical not (see Note 9)	S 5/6
‡	Broken bar (see Note 9)	S 5/7
▧	Diagonal (see Notes 1, 2, 8)	S 5/8
▨	Reverse diagonal (see Notes 1, 3, 8)	S 5/9
▩	Filled diagonal (see Notes 1, 4, 8)	S 5/10

**Table D.25/T.101 – Miscellaneous symbols (cont.)**

Graphic	Name or description	Coded representation
	Filled reverse diagonal (see Notes 1, 5, 8)	S 5/11
	Cross (see Notes 1, 6, 8)	S 6/5
–	Horizontal bar (see Notes 1, 7, 8)	S 5/0

NOTES

- 1 The use of these characters are discouraged and are retained only to allow upward compatibility from older implementations.
- 2 The diagonal extends from the lower left corner of the character field to the upper right corner of the character field.
- 3 The reverse diagonal extends from the lower right corner of the character field to the upper left corner of the character field.
- 4 The filled diagonal is a filled triangle occupying the lower right half of the character field.
- 5 The filled reverse diagonal is a filled triangle occupying the lower left half of the character field.
- 6 The cross character consists of a 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 overlaid with a 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.
- 7 The typical graphic representation of the horizontal bar is a horizontal line about level with the middle of the capital letter.
- 8 The rectangles surrounding the characters described by Notes 2 to 7 above are for illustrative purposes only and are not part of the graphic symbols.
- 9 The representation of these characters as a full horizontal line and a full vertical line, respectively, is permitted to allow upward compatibility from older implementations.

Subclause D.7.3, Table D.27 of Recommendation T.101, Annex D, Data Syntax III, is replaced by the following revised text:

**Table D.27/T.101 – Miscellaneous Non-spacing characters**

Graphic	Name or description	Coded representation
	Non-spacing underline (see Notes 1, 3)	S 4/12
–	Non-spacing vector overbar (see Notes 1, 2)	S 4/0
/	Non-spacing slant (see Notes 1, 3)	S 4/9

NOTES

- 1 The use of these characters is discouraged and they are retained only to allow upward compatibility from older implementations.
- 2 The graphic representation of non-spacing vector overbar is that of a vector arrow just above the top of the capital letter.
- 3 The rectangle surrounding the non-spacing underline is for illustrative purposes only and is not part of the graphic symbol.

## 15.4 NO BREAK SPACE and SOFT HYPHEN

A new subclause D.4.5 is added to Recommendation T.101, Annex D, Data Syntax III, following the existing subclause D.4.4 with the following revised text:

### D.4.5 NO BREAK SPACE and SOFT HYPHEN

NO BREAK SPACE (NBSP) is a special character which has the same presentation attributes as SPACE but which does not cause word wrap since it is not in the list of word wrap characters given in D.6.2.7.11.

SOFT HYPHEN (SHY) is a character which has the same presentation attributes as HYPHEN (see D.7.3, Table D.21) when it is displayed. SOFT HYPHEN is normally not displayed and does not advance the cursor position when it is not displayed. If word wrap occurs upon a SOFT HYPHEN, then the SOFT HYPHEN is displayed just before the wrap condition cause movement of the cursor.

Subclause D.6.2.7.11, paragraph 2, is replaced with the following revised text:

Words consisting entirely of alphabetic characters (see Table D.18) and one or more embedded (i.e. not at the beginning or end of the word special terminating characters (! " \$ % ( ) [ ] < > { } ^ \* + - / , . : ; = ? \_ ~ ) or SHY (see D.4.5) may be broken between the special terminating character (or SHY) and the following character, which causes as much of the word to fit on the current line as possible. All other words shall be kept together on a single line.

## 16 Terminal identification

A common terminal identification mechanism has been added to ITU-T Recommendation T.106. The following clause is added following new clause D.10 to describe the terminal identification message parameters particular to this data syntax.

### D.11 Terminal identification message

A terminal identifier command and response message provides server equipment the capability to enquire of the terminal the facilities supported in the terminal. The response message contains information about both the data syntax and options implemented as well as the input/output facilities supported in the terminal.

A terminal identification is requested by a Videotex Access Point or server by sending the following sequence to the terminal:

1/15 2/0 4/0

In response to the terminal identification request, a response sequence of the following form should be transmitted back to the server:

1/15 2/0 x/y ... x/y 4/0 0/13

The CARRIAGE RETURN character 0/13 provides data forwarding over packet network X.28 interfaces, and is optional. The intermediate codes represent the capabilities of the terminal and are specified in Recommendation T.106. Certain codes pertain to this Data Syntax and are listed below:

*Code*            *Description*

1/5 2/0 = terminal identification response introducer – which is to be followed by:

5/4 = data syntax description introducer – which is to be followed by:

data syntax identifier:

3/1 = Data Syntax I

3/2 = Data Syntax II

3/3 = Data Syntax III

3/4 = reserved

:

3/14 = private

which is to be followed by the data syntax profile and facilities identification: the profile and facilities identification consists of values in the range 4/1-4/15 and 6/0-7/15, terminated by the code 3/15. The codes following a particular data syntax identification pertain to that particular data syntax.

The data syntax identifier sequence is to be terminated by:

3/15 = data syntax identifier termination code

the terminal identification response is to be terminated by:

4/0 = response message terminator code

response message for a terminal supporting Data Syntax III would therefore have the form:

1/15 2/0 ...

5/4 3/3 x/y ... x/y 3/15

...

4/0 0/13

where x/y consist of characters chosen from columns 4/1-4/15 and 6/0-7/15 and identify the particular capabilities of the Data Syntax III terminal.

For the identification of I/O device facilities on a Data Syntax III terminal, the following codes are defined:

4/1	Display output Device
4/2	Peripheral Port Output Device (nominally RS 232)
4/3	Printer Device (nominally on a separate I/O port)
4/4-4/12	Reserved
4/13	Integrated Smart Card Device
4/14	Peripheral Port Input Device (nominally RS 232)
4/15	Keyboard Device

For the identification of other terminal facilities on a Data Syntax III terminal, the following codes are used:

6/1	Support of a Banking Security Encryption Option
6/2-7/5	Reserved

## Annex E

### Audio data syntax

#### E.1 Introduction

This annex defines a data syntax for conveying audio data in a Videtotex environment. In this data syntax a variety of audio encoding techniques are embedded in one general structure. In this specification no algorithms nor specific sound encodings are specified. It allows for the embedding of both waveform and phonemic encodings.

The specification closely follows the concepts and coding techniques as defined in ISO/IEC 9281 [8] for the identification of pictorial information and for switching between picture coding environments and coding systems according to ISO 2022 [7].

## E.2 Scope

This annex specifies the data syntax to be used by Videtotex services for conveying sound information.

This annex is applicable to terminals, connected to public data networks. Typically, these will be terminals, supporting ISDN Syntax-based Videtotex, to be attached at either side of a T-reference point or coincident S-and T-reference points of a public ISDN.

## E.3 Normative references

- [1] CCITT Recommendation G.711 (1988), *Pulse Code Modulation of voice frequencies*.
- [2] CCITT Recommendation G.721 (1988), *32 kbit/s adaptive differential pulse code modulation*.
- [3] CCITT Recommendation G.722 (1988), *7 kHz audio-coding within 64 kbit/s*.
- [4] CCITT Recommendation G.723 (1988), *Extensions of Recommendation G.721 ADPCM to 24 and 40 kbit/s for DCME application*.
- [5] CCITT Recommendation J.41 (1988), *Characteristics of equipment for the coding of analogue high quality sound program signals for transmission on 384 kbit/s channels*.
- [6] CCITT Recommendation J.42 (1988), *Characteristics of equipment for the coding of analogue medium quality sound program signals for transmission on 384 kbit/s channels*.
- [7] ISO 2022, *Information Processing – ISO 7-bit and 8-bit coded character sets – Code extension techniques*.
- [8] ISO/IEC 9281, *Information technology – Picture coding methods – Part I: Identification*.
- [9] I-ETS 300 036, *European digital cellular telecommunications system (phase I ); Full-rate speech transcoding (GSM 06-10)*.
- [10] ISO CD 11172-3, *Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s – Audio part*.

## E.4 Definitions

**E.4.1 audio bit rate:** The bit rate required to convey data which is coded according to a specific encoding in real time.

**E.4.2 encoding:** Coding method of audio data according to a specific algorithm and coding convention.

**E.4.3 recording level:** The mean level of the sound signal as it was recorded.

**E.4.4 sampling rate:** Aspect of a sound encoding defining the amount of samples encoded per time unit.

**E.4.5 synchronization mode:** Attribute of an encoding defining the synchronization to be applied to the audio data.

**E.4.6 transfer rate:** The effective amount of data bits which can be exchanged between communicating entities.

**E.4.7 translation mode:** Attribute of an encoding defining the method of packing data into octets in order to achieve transparency.

## **E.5 Symbols and abbreviations**

ADPCM	Adaptive Differential Pulse Code Modulation
CMI	Coding Method Identifier
LI	Length Indicator
PCD	Picture Coding Delimiter
PCE	Picture Control Entity
PDE	Picture Data Entity
PE	Picture Entity
PI	Picture Identifier
PCM	Pulse Code Modulation
PM	Picture Mode
RPE-LTP	Residual Pulse Excitation – Long Term Predictive
VPCE	Videotex Presentation Control Element

## **E.6 Overview**

The data syntax allows for the embedding of a variety of different audio coding techniques in one single overall structure. Each information element is tagged with an introducer indicating:

- the coding technique being used;
- the audio bit rate of the coding.

The audio bit rate indicates the bit rate, with which the used encoding can be conveyed in real time. The audio bit rate is strongly related to the sampling rate and may be different from the actual transfer rate of the protocol being used for data exchange.

The data syntax provides for block wise transmission of audio data, thus avoiding the need for terminals to scan through streams of audio data for delimiter sequences.

To increase the actual data throughput, the data syntax provides for transparent data transport, thereby surpassing the coding convention, that data representing information is coded in columns 2-7 (and 10-15 in an 8-bit environment).

## **E.7 Introducer**

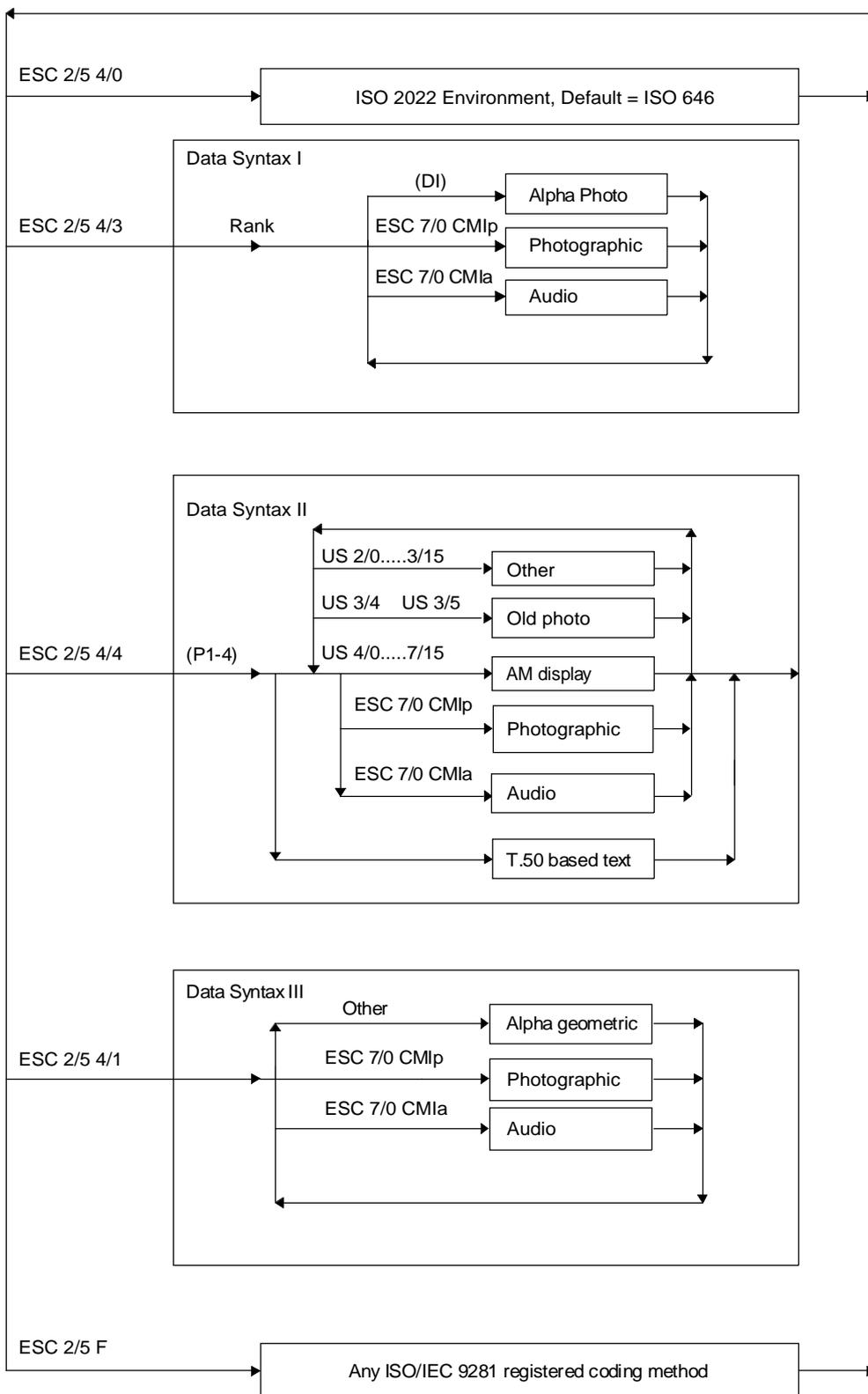
The introducer for sound is in accordance with ISO/IEC 9281 [8] and has the following coding: ESC 7/0 2/4 where 2/4 identifies audio coding.

The return from the sound environment is done by using ESC 2/5 4/0 (return to default ISO 2022 environment) or by switching explicitly to an identified environment using another ESC 2/5 x/y sequence, or by the end of an ISO/IEC 9281 [8] picture element, at which time the current ISO 2022 [7] coding environment (i.e. the base Videotex data syntax) takes effect.

## **E.8 ISO/IEC 9281 [8] syntax and switching structure**

### **E.8.1 Overall switching of coding environment**

ISO/IEC 9281 [8] describes a technique for identifying coding methods. The Videotex audio mode is one of the coding methods identified by ISO/IEC 9281 [8]. The diagram in Figure E.1 gives an overview of the relationship between the Videotex data syntaxes and ISO/IEC 9281 [8] coding environments.



T0823230-95/d267

- |    |                           |      |  |
|----|---------------------------|------|--|
| AM | Alpha Mosaic              | CMla | Any CMI for Videotex audio data        |
| Di | Data Syntax I specific    | CMlp | Any CMI for Videotex photographic data |
| P  | Profile in Data Syntax II | Rank | Data Syntax I specific                 |
| F  | Final code                |      |  |

Figure E.1/T.101 – Global switching mechanism



The Length Indicator follows the conventions of ISO/IEC 9281 [8] and consists of a series of 1 or more octets, which contain 6 bits per octet to encode the length. All but the last octet have bit 6 set to 1. The most significant bit (bit 8) is reserved for parity.

The Picture Data Entity contains, depending on the first octet of the PDE value, either header information or audio data coded according to one of the techniques as defined in chapter encoding.

NOTE 2 – In some Videotex systems, in a situation where an interruption of the audio data flow is required (cause probably by user interaction) the following may occur. The PE being sent to the terminal is completely sent. The next PE sent to the terminal can have the PDE value set to 04/02 to indicate the end of the data, the PDE will contain only one byte (Length Indicator = 1.)

NOTE 3 – If possible, the speed of the network should be taken into account when defining the number of bytes in a PDE with the aim of an interaction response time tolerable. As a rough guide a block length that does not result in a response time exceeding 200-500 milliseconds may be used.

## E.9 Sound header

### E.9.1 Introduction

The sound header indicates the following attributes of subsequent sound blocks.

- the encoding;
- the bitrate, used to encode the sound informations;
- the recording level;
- the translation mode;
- the synchronization mode.

The coding structure allows for the omission of any of these attributes in a particular sound header. When omitted, an attribute shall take the value of the last appearance in a sound header or when never defined by a default value.

### E.9.2 Header structure

#### E.9.2.1 Concepts

The coding structure allocated columns 2-6 of the code table to the attributes mentioned in the previous subclause, thereby providing the possibility to omit particular attributes in a sound header, as defined in Table E.1.

**Table E.1/T.101 – Code table mapping**

Encoding	Column 2
Bitrate	Column 3
Recording level	Column 4
Translation mode	Column 5
Synchronization mode	Column 6
NOTE – The use of column 7 is for further study. Any octets in the sound header which do not pertain to columns 2-6 shall be ignored.	

#### E.9.2.2 Encoding

The encoding parameter shall take one of the following values:

- 02/00: PCM A Law (CCITT G.711 [0])
- 02/01: PCM mu Law (CCITT G.711 [0])
- 02/02: ADPCM (CCITT G.721 [2]/G.723 [4])
- 02/03: Subband ADPCM (CCITT G.722 [3])

- 02/04: RPE-LPT coding method (ETSI I-ETS 300 036 [9])
- 02/05: near instantaneous (CCITT J.41 [5])
- 02/06: subband ADPCM (CCITT J.42 [6])
- 02/07: MPEG Audio (ISO CD 11172-3 [10])
- 02/14: private

The values 02/08 until 02/13 are reserved and should be ignored when received. Value 02/15 is reserved for future extension mechanisms and should be ignored when received.

### **E.9.2.3 Bitrate**

The bitrate parameter shall take one of the following values:

- 03/00: 8 kbit/s
- 03/01: 16 kbit/s
- 03/02: 24 kbit/s
- 03/03: 32 kbit/s
- 03/04: 40 kbit/s
- 03/05: 48 kbit/s
- 03/06: 56 kbit/s
- 03/07: 64 kbit/s
- 03/08: 13 kbit/s (GSM encoding [9])
- 03/09: 2,4 kbit/s (provisional)
- 03/10: 4,8 kbit/s (provisional)
- 03/11: 128 kbit/s (provisional)
- 03/12: 192 kbit/s (provisional)
- 03/13: 384 kbit/s (provisional)
- 03/14: 256 kbit/s (provisional)

Value 03/15 is reserved for future extension mechanisms and should be ignored when received.

### **E.9.2.4 Recording level**

The volume parameter shall take one of the following values:

- 4/0: Recording level not defined by the source; the terminal may adapt the output level by itself.

The values from 4/1 until 4/14 are reserved and shall be ignored when received. Value 4/15 is reserved for future extension mechanism and shall be ignored when received.

### **E.9.2.5 Translation mode**

The translation mode parameter shall take one of the following values:

- 05/00: translation mode 4 (shift 7 bits)
- 05/01: translation mode 3 (shift 8 bits)
- 05/02: translation mode 2 (3 in 4 encoding)
- 05/03: translation mode 1 (no translation except for US character)
- 05/04: translation mode 0 (no translation)
- 05/05: translation mode 5 (no translation except specific characters)

For a description of translation modes 0 until 5 refer to E.12.

The values 05/06 until 05/14 are reserved and should be ignored when received. Value 05/15 is reserved for future extension mechanisms and should be ignored when received.

**E.9.2.6 Synchronization mode**

The synchronization mode parameter shall take one of the following values:

06/00: output audio as soon as possible.

The values 06/01 until 06/14 are reserved and shall be ignored when received. Value 06/15 is reserved for future extension mechanisms and shall be ignored when received.

**E.9.3 Default values**

The default values of the parameters are given in Table E.2.

**Table E.2/T.101 – Default values**

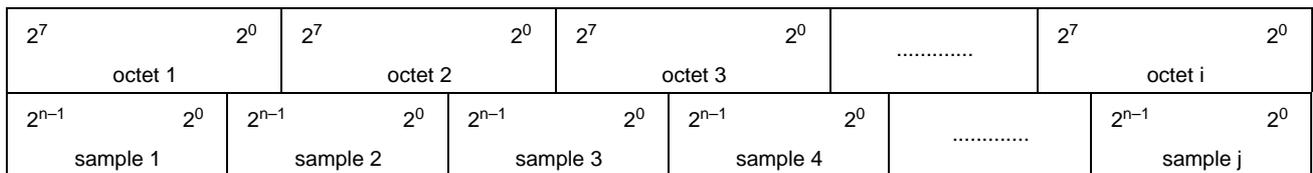
Encoding	02/02 (ADPCM)
Bitrate	03/03 (32 kbit/s)
Recording level	04/00 (not defined by the source)
Translation mode	05/04 (no translation)
Synchronization mode	06/00 «Presentation as soon as possible»

**E.10 Sound block**

A sound block is coded conform the encoding specified in the previous sound header or, if absent, conform the default algorithm.

Several Recommendations specified in the list of encodings define both a framing structure and a sound encoding. This framing structure specifies the multiplexing of sound data with other (e.g. control or display) data.

A sound block shall only use the encoding of sound data specified in these Recommendations. If sound samples are encoded using less than 8 bits, then the data bits shall be concatenated according to the following structure to obtain the data stream (prior to applying the transparency mechanism) as shown in Figure E.2.



**Figure E.2/T.101 – Sampling in less than 8 bits**

For CCITT Recommendation G.721 [2] sound, the sample is as in CCITT Recommendation G.721 [2]:  $2^{n-1}$  is  $I_1$  and  $2^0$  is  $I_4$ .

For CCITT Recommendation G.722 [3] sound at 56 kbit/s, the sample is as in CCITT Recommendation G.722 [3]:  $2^{n-1}$  is  $I_{H1}$  and  $2^0$  is  $I_{L5}$ .

For CCITT Recommendation G.722 [3] sound at 48 kbit/s, the sample is as in CCITT Recommendation G.722 [3]:  $2^{n-1}$  is  $I_{H1}$  and  $2^0$  is  $I_{L4}$ .

NOTE –  $I_1$  and  $I_4$  are defined in CCITT Recommendation G.721 [2].  $I_{H1}$ ,  $I_{L4}$  and  $I_{L5}$  are defined in CCITT Recommendation G.722 [3].

## E.11 Application rules for ISDN syntax-based Videotex

### E.11.1 Encoding/bitrate combinations

The algorithms which are applicable to ISDN syntax-based Videotex are given in Table E.3 which indicates also whether to support them is mandatory or not.

**Table E.3/T.101 – Applicable algorithms**

Encoding	Bitrate
ADPCM	24 kbit/s (optional)
ADPCM	32 kbit/s (mandatory)
ADPCM	40 kbit/s (optional)
Subband ADPCM	48 kbit/s (optional)
Subband ADPCM	56 kbit/s (optional)
RPE-LTP coding	13 kbit/s (optional)

### E.11.2 Translation mechanisms

The following translation mechanisms shall only be used in the case of ISDN syntax-based Videotex.

- mode 0 (Mandatory);
- mode 2 (see Note).

NOTE – The mode 2 is required in all cases, except for close systems (including external data bases) which guarantee 8-bit data transparency.

## E.12 Translation modes

### E.12.1 Mode 0 (no translation is performed)

### E.12.2 Mode 1 (No translation except US)

Under this scheme, no translation of data is performed, except that all US (1/15) characters in the Sound Block data are represented by two contiguous US characters in the transmitted data stream.

### E.12.3 Mode 2 (3-in-4 coding)

Each group of three bytes in the Sound Block data is mapped into four bytes for transmission, as shown in Table E.4. Any remaining group of one or two bytes at the end of a block of data is mapped into two or three bytes respectively, with undefined bits set to zero.

**Table E.4/T.101 – 3-in-4 Coding scheme**

Sound block	3 bytes	3 bytes	1, 2 or 3 bytes
Transmitted	4 bytes	4 bytes	2, 3 or 4 bytes

Within each group, the bits are mapped as follows, where “bxy” denotes bit y of byte x in the user data. Bit 7 is not taken into account by this scheme, but may be determined by characteristics of the transmission path in use (for example, if parity is required).

a) *Three bytes of Block Sound data*

Transmitted	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
1st byte	X	1	b <sub>17</sub>	b <sub>16</sub>	b <sub>27</sub>	b <sub>26</sub>	b <sub>37</sub>	b <sub>36</sub>
2nd byte	X	1	b <sub>15</sub>	b <sub>14</sub>	b <sub>13</sub>	b <sub>12</sub>	b <sub>11</sub>	b <sub>10</sub>
3rd byte	X	1	b <sub>25</sub>	b <sub>24</sub>	b <sub>23</sub>	b <sub>22</sub>	b <sub>21</sub>	b <sub>20</sub>
4th byte	X	1	b <sub>35</sub>	b <sub>34</sub>	b <sub>33</sub>	b <sub>32</sub>	b <sub>31</sub>	b <sub>30</sub>

b) *Two bytes of Block sound data at end of sequence*

Transmitted	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
1st byte	X	1	b <sub>17</sub>	b <sub>16</sub>	b <sub>27</sub>	b <sub>26</sub>	0	0
2nd byte	X	1	b <sub>15</sub>	b <sub>14</sub>	b <sub>13</sub>	b <sub>12</sub>	b <sub>11</sub>	b <sub>10</sub>
3rd byte	X	1	b <sub>25</sub>	b <sub>24</sub>	b <sub>23</sub>	b <sub>22</sub>	b <sub>21</sub>	b <sub>20</sub>

c) *One byte of Block Sound data at end of sequence*

Transmitted	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
1st byte	X	1	b <sub>17</sub>	b <sub>16</sub>	0	0	0	0
2nd byte	X	1	b <sub>15</sub>	b <sub>14</sub>	b <sub>13</sub>	b <sub>12</sub>	b <sub>11</sub>	b <sub>10</sub>

**E.12.4 Mode 3 (Shift scheme – 8 bits)**

In this scheme, bytes of Sound Block data are each mapped into one of two bytes of transmitted data, as shown in Table E.5; in mode 3, the most significant bit of each transmitted byte is taken into account. Note that most of the conversions are optional. Either entity should be prepared to accept any mixture of converted or unconverted data in these cases.

**Table E.5/T.101 – Code conversion for 8-bits shift scheme (mode 3)**

Sound block	Conversion	O/M	Transmitted
0/0-0/12	$7/14 x + 5/0$	O	7/14, 5/0-5/12
0/13	$7/14 x + 5/0$	M	7/14, 5/13
0/14-01/14	$7/14 x + 5/0$	O	7/14, 5/14-6/14
1/15	$7/14 x + 5/0$	M	7/14, 6/15
2/0	7/13	O	7/13
2/1-7/10	None	–	2/1-7/10
7/11	$7/11 x - 5/8$	M	7/11, 2/3
7/12, 7/13	$7/11 x - 5/8$	M	7/11, 2/4-2/5
7/14	$7/11 x - 5/8$	M	7/11, 2/6
7/15	$7/11 x - 5/8$	M	7/11, 2/7
8/0-13/0	$7/11 x - 5/8$	O	7/11, 2/8-7/8
13/1-15/15	$7/14 x - 5/0$	O	7/14, 2/1-4/15
O Optional			
M Mandatory			

### E.12.5 Mode 4 (Shift scheme – 7 bits)

In this scheme, bytes of Sound Block data are each mapped into one or two bytes of transmitted data, as shown in Table E.6; in mode 4, the most significant bit of each transmitted byte is not taken into account. Note that most of the conversions are optional. Either entity should be prepared to accept any mixture of converted or unconverted data in these cases.

**Table E.6/T.101 – Code conversion for 7-bits shift scheme (mode 4)**

Sound Block	Sender's Conversion	O/M	Transmitted Data
0/0-1/14	7/14, x + 80 (X + 50 hex)	O	7/14, 5/0-6/14
1/15	7/14, 6/15	M	7/14, 6/15
2/0	7/13	O	7/13
2/1-7/10	None	–	2/1-7/10
7/11-7/15	7/11, x – 88 (X – 58 hex)	M	7/11, 2/3-2/7
8/0-13/0	7/11, x – 88 (X – 58 hex)	M	7/11, 2/8-7/8
13/1-15/15	7/14, x – 80 (X + 50 hex)	M	7/14, 2/1-4/15
– Irrelevant O Optional M Mandatory NOTE – In mode 4 the transmitted bytes may have the most significant bit set.			

### E.12.6 Mode 5 (No translation except specific characters)

Under this scheme no translation of data is performed except that all of the codes in the photographic data stream identified in the list below are replaced by the code DLE (1/0) followed by the replacement code identified in the list below:

<i>Code</i>	<i>Replacement Code</i>
0/1	4/1
0/2	4/2
0/3	4/3
0/4	4/4
0/5	4/5
0/6	4/6
0/7	4/7
1/0	5/0
1/1	5/1
1/2	5/2
1/3	5/3
1/4	5/4
1/5	5/5
1/6	5/6
1/7	5/7

1/8	5/8
1/9	5/9
1/10	5/10
1/11	5/11
1/15	5/15

NOTE – This mode 5 is intended to be used in an 8-bit environment but is not usable with transmission procedures which use as control codes two bytes sequences starting with DLE.

## Annex F

### Photographic data syntax

#### Introduction

This annex is part of a series of Recommendations which describe the Videotex presentation layer data syntax.

This annex defines a data syntax to be used for conveying photographic data in a Videotex environment. The necessary tools are provided for the transfer of photographic data, typically from a Videotex Host to a Videotex terminal. This data syntax is equally applicable to either storage or communication applications and is independent of physical device or transmission media.

This annex does not deal with the visible appearance of the displayed pictures, however all the necessary source image information is provided to make the proper physical adaptation at the receiving side. The specification of post-processing techniques is left to the implementors and is, therefore, outside the scope of this annex.

More precisely, this annex defines the syntax and semantics of image data and image attributes for photographic Videotex interchange purposes. In particular, it addresses the various aspects of image dimensionality such as spatial, amplitudinal, temporal and spectral content, it provides some basic tools for positioning photographic images within a defined area, it also addresses the structure and organisation of the data and uses standardized compression schemes. In particular, the ISO-Joint Photographic Experts Group (JPEG) compression algorithm [13], based on the Discrete Cosine Transform (DCT), the facsimile ITU-T Recommendation T.4 [17] and CCITT Recommendation T.6 [18] coding algorithms are used. In this annex the algorithms or compression techniques themselves are not described, references are provided.

The intention of this annex is primarily to provide Videotex application developers with a sufficient set of photographic transfer tools which are independent of the equipment used to implement/provide them. This annex is intended to support operations on and display of, various classes of images from a wide variety of imaging applications. However, to ensure that compatibility can be achieved between various Videotex services supporting photographic mode, some realistic and specific characteristics are chosen and defined in the clause on profiles (clause 11). In the future, other selections might be made allowing the definition of new recommended profiles.

This annex closely follows the concepts and coding techniques as described in ISO/IEC 9281, Part 1 [11] for the identification of pictorial information and for switching between picture environments and coding systems according to ISO 2022 [10].

#### F.1 Scope

This annex specifies the data syntax to be used by Videotex services for conveying photographic data.

In general, it applies to the interchange of photographic data via storage or transmission media.

This annex is applicable to Videotex terminals connected Packet to various types of telecommunication networks including:

- a Public Switched Telephone Network (PSTN);
- a Packet Switched Public Data Network (PSPDN); or
- an Integrated Services Digital Network (ISDN).

For the ISDN case, these terminals will typically support “ISDN Syntax-based Videotex” (ITU-T Recommendation T.105 [3]).

The syntax allows for some private extensions beyond the transmission of still pictures. For example, a provision has been made for the transmission of a “difference” image to allow a slow scan television type of application.

## F.2 Normative references

This annex incorporates by dated or undated reference, provisions from other publications. These normative references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references subsequent amendments to or revisions of, any of these publications apply to this annex only when incorporated in it by amendment or revision. For undated references the latest edition of the publication referred to applies.

- [1] ITU-T Recommendation T.101, Annexes B, C and D.
- [2] ITU-T Recommendation T.101, section “TFI”, subclause on Code extension.
- [3] ITU-T Recommendation T.105, *Syntax-based Videotex, End-to-end protocols*.
- [4] CCITT Recommendation F.300 (1988), *Videotex service*.
- [5] ITU-T Recommendation T.101, *International interworking for Videotex*.
- [6] CCITT Recommendation H.261 (1988), *Common intermediate format*.
- [7] ITU-T Recommendation T.51, *Coded Character sets for Telematic services*.
- [8] CCITT Recommendation T.61 (1988), *Character repertoire and coded character sets for the international teletex service*.
- [9] CCIR Recommendation 601-1 (1986), *Encoding Parameters of Digital Television For Studios*.
- [10] ISO 2022 (1986), *Information Processing – ISO 7-bit and 8-bit coded character sets – Code extension techniques*.
- [11] ISO/IEC 9281-1 (1990), *Information technology – Picture coding methods – Part 1: Identification*.
- [12] ISO/IEC 9281-2 (1990), *Information Technology – Picture coding methods – Part 2: Procedure for registration*.
- [13] ITU-T Rec. T.81 | ISO/IEC 10918-1, *Digital compression and coding of continuous-tone images*.
- [14] ITU-T Recommendation T.50, *International Reference Alphabet*
- [15] ISO 6937 (1991), *Information processing – Coded character sets for text communication*.
- [16] ISO 2375 (1991), *Data Processing – Procedure for registration of escape sequences*.
- [17] ITU-T Recommendation T.4 (1992), *Standardization of group 3 facsimile apparatus for document transmission*.
- [18] CCITT Recommendation T.6 (1988), *Facsimile coding schemes and coding control functions for group 4 facsimile apparatus*.
- [19] ITU-T Recommendation T.82 (1992), *Coded representation of picture and audio information – Progressive bi-level image compression*.
- [20] ITU-T Recommendation T.30 (1992), *Procedures for document facsimile transmission in the general switched telephone network*.
- [21] ITU-T Recommendation T.563 (1992), *Terminal Characteristics for Group 4 facsimile apparatus*.

## **F.3 Definitions and abbreviations**

### **F.3.1. Definitions**

For the purpose of this annex the following definitions apply.

**F.3.1.1 aspect ratio:** The ratio of the width to the height of a rectangular area, such as the defined display area.

**F.3.1.2 attribute:** A particular property or quantity defined in this syntax and described by a number of parameters (e.g. the source picture specifications).

**F.3.1.3 baseline:** The basic sequential DCT-based encoding and decoding process specified in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13].

**F.3.1.4 continuous tone image:** An image comprised of data which exhibits a first order continuity in the analogue domain and requires, when digitized, more than one bit to describe each sample contained in one or more of its components [monochrome (grey scale) or colour pictures] e.g. a monochrome image needs at least 6 bits/picture element (64 grey levels) to appear “continuous” to the eye.

**F.3.1.5 data syntax I:** Term used within ITU-T for one of the recommended world-wide Videotex data syntaxes originating from the Japanese Character And Pattern Telephone Access Information Network (CAPTAIN) system.

**F.3.1.6 data syntax II:** Term used within ITU-T for one of the recommended world-wide Videotex data syntaxes originating from the European CEPT Videotex syntax.

**F.3.1.7 data syntax III:** Term used within ITU-T for one of the recommended world-wide Videotex data syntaxes originating from the North American Presentation Layer Protocol Syntax (NAPLPS).

**F.3.1.8 defined display area (DDA), Physical [Physical Defined Display Area (DDA)]:** A rectangular area of the full screen area where photographic data, text etc. shall be displayed.

**F.3.1.9 defined display area (DDA), Logical (Logical DDA):** A unit square, the length of all sides being one unit, co-ordinates being defined as fractions of unity (unit screen concept). The origin is coincident with the bottom left corner of the physical DDA and one side is coincident with the longest side of the physical DDA.

**F.3.1.10 defined Display Area, Source (Source DDA):** The virtual display space where the source image was encoded and which is to be mapped for display either to the full screen area or to the physical DDA.

**F.3.1.11 discrete cosine transformation (DCT):** See ITU-T Rec. T.81 | ISO/IEC 10918-1 [13].

**F.3.1.12 facsimile picture:** A photographic picture, encoded with T.4 [17] or T.6 [18] facsimile coding, using the present Videotex Photographic Syntax.

**F.3.1.13 full screen area:** The part of a display screen where photographic data can be displayed, it normally means a display with no borders.

**F.3.1.14 forward DCT:** See ITU-T Rec. T.81 | ISO/IEC 10918-1 [13].

**F.3.1.15 inverse DCT:** See ITU-T Rec. T.81 | ISO/IEC 10918-1 [13].

**F.3.1.16 image attribute:** The various properties of a continuous tone image described by a number of parameters.

**F.3.1.17 image data:** The data which represents a continuous tone image in digital form, it contains photographic header data and photographic data.

**F.3.1.18 JPEG compression algorithm:** A general term for referring to any one of the possible modes of encoding defined in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13].

**F.3.1.19 normalized co-ordinate:** A co-ordinate specified in a device independent co-ordinate system, normalized to some range (usually to 1).

**F.3.1.20 page:** See Defined Display Area, Source (Source DDA). If digitized with a facsimile scanner a page is generally equivalent to the whole facsimile content of an ISO paper sheet format (ISO A4, B4, A3...)

**F.3.1.21 parameter:** A quantity which is described using one or more sub-parameters.

**F.3.1.22 photographic data:** Pixel based pictorial information usually in compressed digital form; the data includes any tables which are necessary to decode and decompress the data.

**F.3.1.23 photographic data syntax:** The rules by which the photographic header data and the photographic data are formatted.

**F.3.1.24 photographic header data:** Coded data containing the values of the attributes and parameters used for describing the photographic image.

**F.3.1.25 photographic image:** A continuous tone image, e.g. an image represented with 256 shades of grey.

**F.3.1.26 photographic mode:** The mode of operation of a Videotex terminal while it is receiving photographic header data and photographic data.

**F.3.1.27 photographic profile:** A collection of attributes with parameters set to a given value to represent a type of source image and define a mode of photographic image coding and photographic image transfer.

**F.3.1.28 photo Videotex:** Neologism used for Videotex photographic mode.

**F.3.1.29 physical device:** Any tangible piece of equipment (e.g. personal computer, display monitor, etc.).

**F.3.1.30 pixel, picture element:** It is the minimum displayable element of an image (see ITU-T Rec. T.81 | ISO/IEC 10918-1 [13]).

**F.3.1.31 pixel density:** Expresses the number of pixels per physical unit (e.g. pixels/mm) in the horizontal and vertical directions.

**F.3.1.32 pixel density ratio:** Ratio between the pixel density in the horizontal direction and the pixel density in the vertical direction.

**F.3.1.33 post-processing technique:** Image processing which is performed (e.g. for display) after the image has been decoded and decompressed.

**F.3.1.34 spatial resolution:** Definition of the size of the image, expressed in the number of pixels per horizontal line and the number of lines per image.

**F.3.1.35 storage media:** A type of physical means to store data.

**F.3.1.36 sub-parameter:** A quantity to which a value can be assigned.

NOTE – Example of the use of attribute, parameter and sub-parameter. Consider the ISDN, it has the following *attributes*, it is digital and supports data transmission with a speed of 64 kbit/s. For the ISDN the *parameter* network speed is assigned the value 64 kbit/s. Two *sub-parameters* can represent this quantity, “numerical speed” i.e. 64 and “unit of measure” i.e. kbit/s.

**F.3.1.37 spectral content:** A physical quantity that measures the frequency content, i.e. the amplitude and phase of each frequency contained in a given physical item. It generally refers to the fourier analysis. For the Discrete Cosine Transformation (DCT) it relates to the amplitude of each DCT basic function (Discrete Cosine) or sub-image. In simple terms, for the image, it gives an idea on the “level of detail” of the source image.

**F.3.1.38 transmission media:** The type of physical means to transport data (e.g. coaxial cable, optical fibre, radio link).

**F.3.1.39 Videotex system:** Text communication system, hardware and software, with the capability of running a Videotex service or application.

**F.3.1.40 Videotex application:** See ITU-T Recommendation F.300 [4].

**F.3.1.41 Videotex host computer:** see ITU-T Recommendation F.300 [4].

**F.3.1.42 Videotex service:** see ITU-T Recommendation F.300 [4].

**F.3.1.43 Videotex terminal:** see ITU-T Recommendation F.300 [4].

NOTE – Reference should also be made to ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] for definitions of terms related to image coding which are used in this Annex but not included in this Clause.

## F.3.2 Abbreviations

For the purpose of this annex the following abbreviations and symbols apply.

AM	Alphamosaic
$C_B$	Chrominance colour difference, Blue
$C_R$	Chrominance colour difference, Red
CCIR	International Radio Consultative Committee
CCITT	International Telegraph and Telephone Consultative Committee
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
CEPT	Commission Européenne des Postes et Télécommunications
CIF	Common intermediate format
CMI	Coding method identifier
CMYK	Cyan, magenta, yellow, black
DCT	Discrete cosine transformation
DDA	Defined display area
DPCM	Differential pulse code modulation
$E'_R, E'_G, E'_B$	Primary (analogue) signals (red, green, blue)
ESC	EScape
ETSI	European Telecommunications Standards Institute
F	“Final character” registered by ISO 2022 registration authority
HDTV	High definition television
IEC	International Electrotechnical Committee
ISDN	Integrated services digital network
ISO	Organization for International Standardization
JPEG	Joint Photographic Experts Group
LI	Length indicator
LSB	Less significant bit
MSB	Most significant bit
P1 to P5	Compatible photographic profiles, numbered 1 to 5
$P_{priv}$	Private photographic profile
PCD	Picture coding delimiter
PCE	Picture control entity
PDDA	Physical defined display area
PDE	Picture data entity

PE	Picture entity
PI	Picture identifier
PM	Picture mode
PPI	Pixel or picture element per 25.4 millimetres
PPM	Pixel per millimetre, or picture element per millimetre
PSPDN	Packet switched public data network
PSTN	Public switched telephone network
QCIF	Quarter of common intermediate format
RGB	Red, green, blue
SDDA	Source defined display area
TFI	Terminal facility identifier
US	Unit separator
VPCE	Videotex presentation control element
Y	Luminance

## F.4 Overview

The photographic data syntax allows for the transmission and display of an image consisting of individually defined Picture elements (Pixels) from two to many grey/colour levels. Digital signal processing techniques are used to compress the image for storage and transmission.

The photographic data syntax allows, in principle, the specification of a wide variety of different modes of photographic images for use in Videotex systems. However, to cater for the foreseen requirements of users and to aid compatibility, some recommended application profiles are specified, they are based on CCIR Recommendation 601, Part 1 [9] and CCITT Recommendation H.261 [6], which describes the Common Intermediate Format (CIF).

This photographic data syntax is basically unidirectional; the photographic data shall be transported from a Videotex Host computer to a Videotex terminal using the syntax.

In Videotex photographic mode the size of a file containing an encoded photographic image can be rather large. One picture may be transmitted in several PEs. The use of several PEs could facilitate the termination of the transmission of a picture by the user.

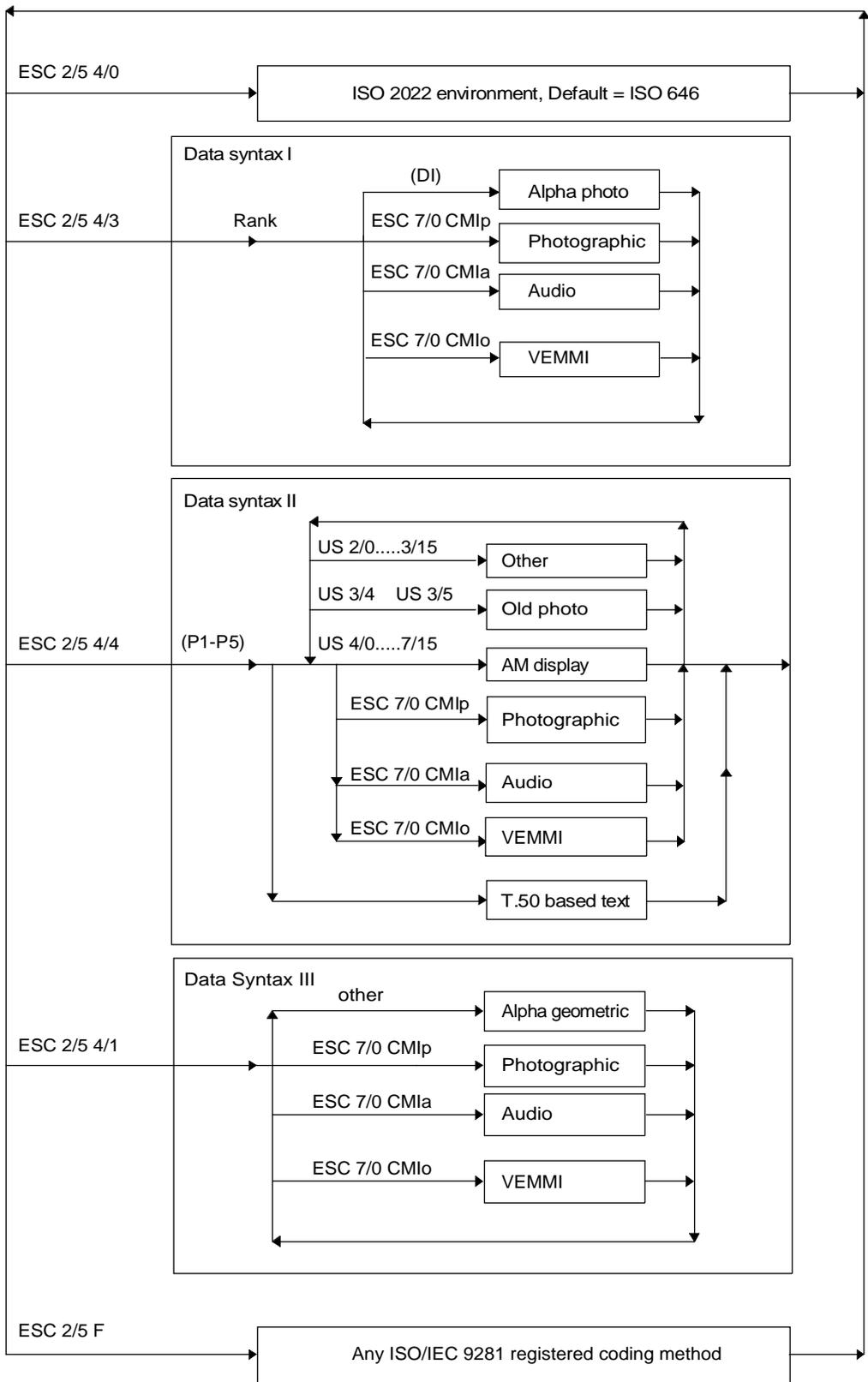
This annex clearly separates the functionalities, the syntax (the formatting rules i.e. structuring the data) and the coding rules (i.e. coding of the header, coding of the photographic data, etc.). The ISO 9281, Part 1 [11] syntax structure used to introduce the photographic header is encoded using 7 bits only. For coding in an 8-bit transfer environment the most significant bit (bit 8) is set to zero.

The photographic data is encoded and represented using 8 bits. For transmission, allowance is made for both a 7-bit and an 8-bit channel.

## F.5 ISO/IEC 9281, Part 1, syntax and switching structure

### F.5.1 Overall switching of coding environment

ISO/IEC 9281, Part 1 [11] describes a technique for identifying picture coding methods. The Videotex photographic mode is one of the picture coding methods identified by ISO/IEC 9281, Part 1 [11]. The diagram in Figure F.1 gives an overview of the relationship between the Videotex data syntaxes and ISO/IEC 9281, Part 1 [11] picture coding environments.



T0817280-94/d268

- |    |   |      |  |
|----|---|------|--|
| DI | Data syntax I specific  | CMIa | Any CMI for Videotex audio data        |
| P  | Profile in data syntax II                                       | CMIp | Any CMI for Videotex photographic data |
| F  | Final code assigned by the ISO 2022 [10] registration authority | Rank | Data syntax I specific                 |

**Figure F.1/T.101 – Global switching mechanism**

From an ISO 2022 [10] environment, a Videotex data syntax can be explicitly entered using an ESC 2/5 F code. This is also the mechanism used for entering from an ISO 2022 environment [10] into an ISO/IEC 9281 [11] environment. The F code (“final byte”) is allocated and registered, according to ISO 2022 [10], by the registration authority ISO 2375 [16]. According to ISO 2375 [16], Annex B, the Videotex data syntaxes are regarded as “coding systems different to that of ISO 2022 [10]”. The F codes are 4/3 for ITU-T data syntax I, 4/4 for ITU-T data syntax II and 4/1 for ITU-T data syntax III.

Since a Videotex terminal usually begins operation, by default, in one of the data syntaxes, it shall not be mandatory to first send an ESC 2/5 F code (F is 4/1, 4/3 or 4/4). The diagram shows how these codes can be used to switch a Videotex terminal supporting more than one data syntax from one data syntax to another.

### F.5.2 Switching into the photographic mode

A Videotex terminal operating within one of the data syntaxes (i.e. a coding system other than that described by ISO 2022 [10]) can enter the ISO/IEC 9281, Part 1 [11] environment of the photographic mode according to their own rules. In the case of Videotex for switching into the ISO/IEC 9281, Part 1 [11] environment of the photographic mode the Picture Code Delimiter (PCD) of the first Picture Element (PE) is used. The Coding Method Identifier (CMI) is used to distinguish between picture coding methods. In the case of Videotex this shall be, for example, a distinction between audio and photographic data.

### F.5.3 ISO/IEC 9281, Part 1, syntax structure

The high level structure of the syntax is as defined in ISO/IEC 9281, Part 1 [11].

In the following description of the syntax 8-bit coding is assumed, thus the word “byte” is used with bit 8 set to zero. The coding described in ISO/IEC 9281, Part 1 [11] is also valid in a 7 bit environment. In this case the word “byte/octet” shall be interpreted as meaning “7 bit byte” and the most significant bit, bit 8, shall not be used.

The structure of the coding is as follows:

```

PE      ::= PCE PDE
PCE     ::= PCD CMI LI
PCD     ::= 01/11 07/00
CMI     ::= PM PI
PM      ::= 02/03 video photo coding (Videotex photographic mode)
PI      ::= <04/00–07/15>
LI24)  ::= x111 1111 <byte 1> <byte 2>....<byte n>
<byte k> ::= x10D DDDD (k = n)
          | x11D DDDD (1 ≤ k < n)

```

x indicates don't care.

D indicates binary number 0 or 1.

Each piece of information, in this case encoded image data, is encoded as one or more Picture Entities (PEs). A PE consists of Picture Control Entity (PCE) which is followed by the actual data packed into a Picture Data Entity (PDE) (see Figure F.2).

<sup>24)</sup> ISO/IEC 9281-1 [11], subclause 5.2.7 should be consulted for a description of the use of the Length Indicator.

PE Picture Entity				
PCE Picture Control Entity				PDE  Picture Data Entity
PCD  Picture Coding Delimiter	CMI Coding Method Identifier		LI  Length Indicator	
	PM Picture Mode	PI Picture Identifier		

**Figure F.2/T.101 – Structure of a Picture Entity**

NOTE 1 – In Videotex photographic mode the size of a file containing an encoded photographic image can be rather large. One picture may be transmitted in several PEs. The use of several PEs could facilitate the termination of the transmission of a picture by the user.

The Picture Control Entity (PCE) consists of a Picture Coding Delimiter (PCD) and a Coding Method Identifier (CMI) followed by a Length Indicator (LI).

The Picture Coding Delimiter (PCD) is a fixed sequence of two octets: 01/11 07/00.

NOTE 2 – 01/11 is ESC.

The Coding Method Identifier (CMI) consists of a Picture Mode (PM) octet, followed by a Picture Identifier octet (PI). For photo Videotex the PM is 02/03 as registered by ISO/IEC 9281, Part 2 [12]. The PI octet specifies the specific coding technique being used. The PI may take any value from the range 04/00-07/15. For the JPEG coding technique 04/00 shall be used.

### F.5.3.1 General use of the Length Indicator (LI)

A length indicator technique is used to specify the number of bytes in the PDE which follows a PCE. The number of bytes is encoded using the LI code as defined in ISO/IEC 9281, Part 1 [11]. Image data can be divided into blocks and sent using a number of PEs. After a LI has expired the terminal shall check all incoming bytes. If transmission of the picture is still in progress the next bytes shall be ESC 7/0 CMI<sub>p</sub> (start of a new photographic PE). If transmission of the picture has terminated the next byte shall be an ESC or US code (data syntax II).

### F.5.3.2 Use of the Picture Identifier (PI) code

The PI shall take one of the values but, at present, it shall only be:

- 04/00 for the ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] JPEG algorithm;
- 04/01 using ITU-T Recommendation T.4 [17] compression method;
- 04/02 using CCITT Recommendation T.6 [18] compression method;
- 04/03 (using ITU-T Rec. T.82 | ISO/IEC CD 11544 compression method [19]).

## F.6 Coding of the Picture Data Entity (PDE)

### F.6.1 Introduction

As described in the previous clause a PE as defined by ISO/IEC 9281, Part 1 [11] consists of a, Picture Control Entity (PCE) followed by a PDE. The coding of the PCE, as it shall be used for the Videotex photographic data syntax, is defined in clause F.5. The PDE contains either photographic header data or photographic data. A mechanism is described below which shall be used for identifying the type of data in a PDE.

**F.6.2 PDE data content identification mechanism**

The first byte after the LI, i.e. the first byte of the PDE (denoted as PDE<sub>1</sub>), identifies the type of information which follows. This byte can have the following values:

- 5/0 for the photographic header data with more photographic header data to follow;
- 5/1 for the last PE of photographic header data;
- 5/2 for the photographic data with more data to follow;
- 5/3 for the last PE of photographic data.

The transmission of the photographic header shall not be interrupted.

The data following the PCE with the first byte of the PDE set to 5/0 is the photographic header data. The photographic header data shall be sent in one or more PEs, the transmission of the photographic header shall not be interrupted. The actual photographic data shall be sent in a number of PEs, with each PE having the PDE<sub>1</sub> value set to 5/2 except the last one which shall have the PDE<sub>1</sub> value set to 5/3. The LI is used in all the PEs (as illustrated in Figure F.3).

1	PCE	PDE 1 = 5/0	Header Data
2	PCE	PDE 1 = 5/1	Header Data
3	PCE	PDE 1 = 5/2	Photographic Data
4	PCE	PDE 1 = 5/2	Photographic Data
5	PCE	PDE 1 = 5/3	Photographic Data

**Example 1** – A complete sequence of PEs (one picture) with the Header Data in two PEs (1 and 2) and the Photographic Data in three PEs (3, 4 and 5).

1	PCE	PDE 1 = 5/1	Header Data
2	PCE	PDE 1 = 5/2	Photographic Data
3	PCE	PDE 1 = 5/2	Photographic Data
4	PCE	PDE 1 = 5/2	Photographic Data
5	PCE, LI = 1	PDE 1 = 5/3	No data

**Example 2** – A complete sequence of PEs (one picture) with the Header Data in one PE (1), the Photographic Data in three PEs (2, 3 and 4). The last PE contains no data.

**Figure F.3/T.101 – Examples for sequences of PEs**

In a situation where an interruption of the photographic image (caused probably by user interaction) one of the two following scenarios shall occur:

#### **Scenario 1:**

In some Videotex systems, the PE being sent to the terminal shall be completely sent. The next PE sent to the terminal shall have the PDE<sub>1</sub> value set to 5/3 to indicate the end of the data. The PDE contains only one byte (length indicator = 1). This solution is preferred.

#### **Scenario 2:**

Where “scenario 1” is not supported by the Videotex system, the terminal shall not require an explicit notification that the transmission of the picture has terminated. The use of the PE with PDE<sub>1</sub> set to 5/3 for aborting the rest shall not be sent. The terminal shall, at the latest, resynchronise on receipt of the next header (5/0).

#### NOTES

- 1 If possible, the speed of the network should be taken into account when defining the number of bytes in a PDE.
- 2 The receipt of the last PE indicated to the terminal by the PDE<sub>1</sub> value 5/3, could be used to indicate to the user that the transmission of the picture has been completed. For example, a message indicating this could be displayed for the user.

A series of pictures with an already stated header may be sent without redefining the attributes.

## **F.7 Photographic header**

### **F.7.1 Introduction**

The photographic header is the specific part of the photographic data syntax that specifies the parameters of the various photographic image attributes that support the needs of photographic Videotex transfer and display. In particular, it shall support various combinations of dimensionality, organization and compression. For example, it could support static bilevel (document) images, static colour [e.g. luminance (Y), chrominance colour difference, blue (C<sub>B</sub>), chrominance colour difference, red (C<sub>R</sub>)], images, iconic images etc., with standardized compression schemes. Extensions for non-standardized use (e.g. animated images) shall be supported.

In the description of the attributes and parameters contained in this clause no indication is given as to whether the parameters are optional or mandatory. In the clause on profiles (clause F.11) the classification of optional or mandatory is assigned to each parameter for each profile.

Some of the parameters of the photographic image attributes may appear to duplicate those already provided in the signalling part of the photographic data ( For example, according to the header of the data stream structure when ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] is used). They are included in the photographic header to give the terminal access to this information without having to decode the encoded data stream. If a parameter is duplicated the values shall be identical, otherwise the result is unpredictable.

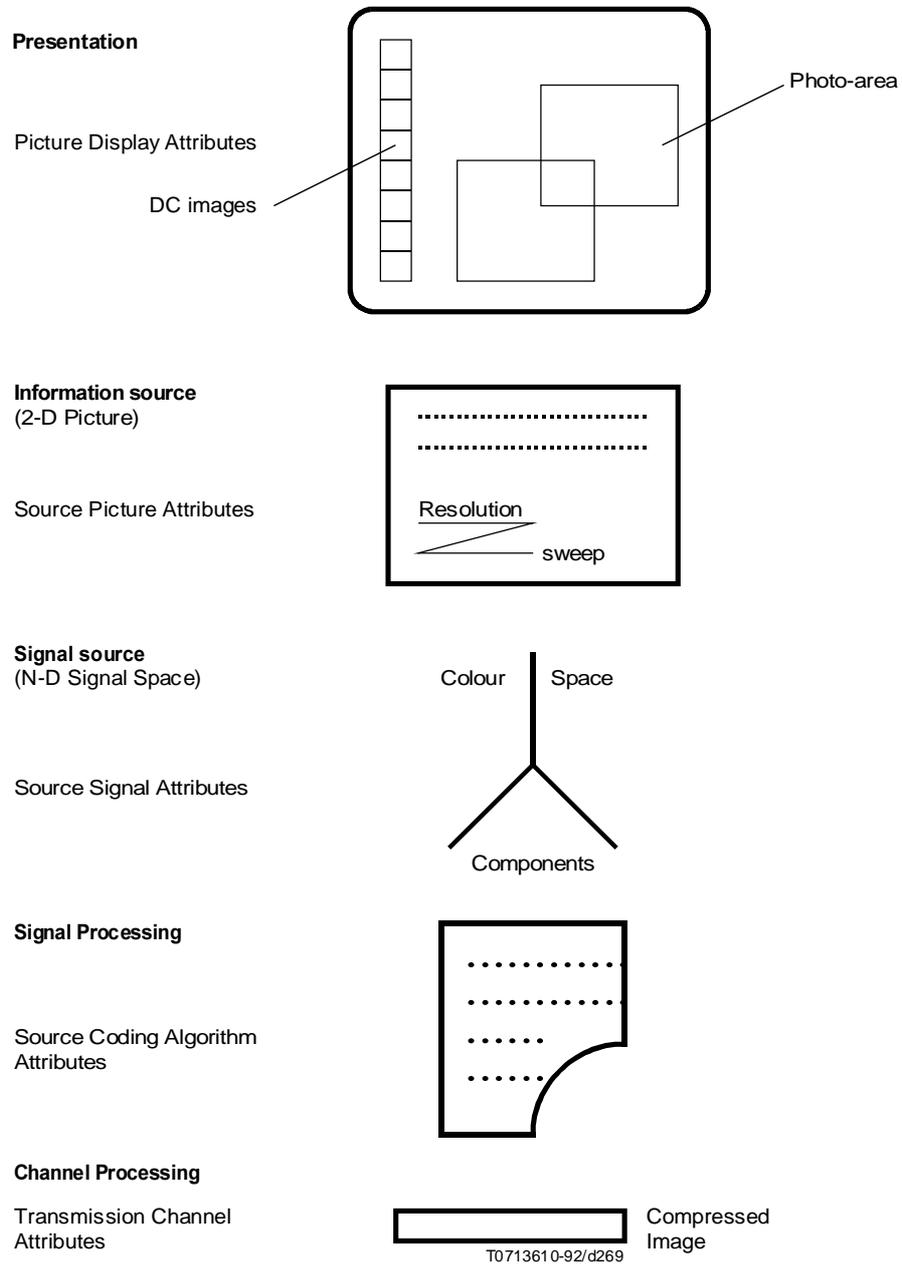
The syntax allows for the omission of some parameters from a particular attribute of the photographic header. Depending upon the value of the parameter status attribute, a parameter shall take either the value of its last appearance in a photographic header or the value assigned as its default. Default values are defined in clause F.10.

In the following text, the header structure is first globally introduced, then each part of the structure is described in terms of attributes. The coding rules used and their application to the defined header shall be described in clause F.8.

### **F.7.2 Header structure**

The general structure of the photographic header is shown in Figure F.4. The structure is composed of the following main attributes:

- parameter status attribute;
- picture display attributes;
- source picture attributes;
- source signal attributes;
- source coding algorithm attributes;
- transmission channel attributes.



**Figure F.4/T.101 – Photographic header structure (excluding the parameter status attribute)**

Table F.1 shows how the various attributes are described using a number of parameters.

**Table F.1/T.101 – Overview of the header functionalities**

Photographic header	
Attributes	Parameters
Parameter Status Attribute <PSA>	Reset to Default <RTD>
Picture Display Attributes <PDA>	Full Screen Display <FSD>
	Source Aspect Ratio <ASR>
	Photo-area Location <LOC>
	Photo-area Size <PAS>
	Picture Placement <PPL>
	Clear Photo-area <CPA>
	Source Picture Attributes <SPA>
	Source Picture Dimensions<PDS>
	Source Pixel Density<PID>
	Source Sweep Direction<SWD>
	DC Images <DCI>
Source Signal Attributes <SSA>	Source Component Description <SCD>
	Source Component Data Precision <CDP>
	Source Component Order <CMO>
	Source Level Assignment <LAS>
Source Coding Algorithm Attributes <SCA>	JPEG Coding Mode <JPG>
	Encoding Table Management <ETM>
	Application Marker Codes Assignment <AMA>
	T4 Coding Mode <T4C>
	T6 Coding Mode <T6C>
Transmission Channel Attributes <TCA>	Translation Mode Encoding <TME>

The selection of the attribute classes follows a “layered structure”, in a manner similar to the OSI-philosophy.

The highest “layer” gives an indication of whether default values should be used. On the “layer” below the picture display attributes are responsible for embedding the transmitted photographic image into the Videotex application environment (i.e. display). On the “layer” below, by using the source picture attributes, definitions are made for how and with which physical properties (such as image size) the physical source picture was captured. On the next “layer” the source signal attributes define in detail the specification of the source signal to be compressed. On the next “layer” the source coding algorithm attributes specify the picture compression technique which was used to create a compressed data stream. Finally, on the lowest “layer”, transmission channel attributes, some specific aspects for the transmission of the compressed data stream are defined (e.g. the use of 7- or 8-bit transparent mode).

### F.7.3 Header functionalities

The name of each attribute and parameter is followed by an abbreviation enclosed in “< >” brackets. The abbreviation is used in the coding clause of this annex (clause F.8) but is introduced here to make cross-referencing between the two clauses easier.

### F.7.3.1 Parameter Status Attribute <PSA>

A single parameter is used to define the status of all parameter values currently in use and the status of the new values being sent. Since this parameter shall always be sent first, to avoid any possible ambiguities, all parameters shall be sent in the order given in clause F.8 (Coding rules).

#### F.7.3.1.1 Reset to default <RTD>

If this parameter is set to “true” all parameter values shall be reset to their default value. If a parameter value is subsequently sent, the value being sent supersedes the default value.

If this parameter is set to “false” all parameter values shall assume the value to which they were set in a previous header. If, as yet, they have not been set they shall assume the default value. If a parameter value is subsequently sent the value sent supersedes the “in use” value.

A single sub-parameter is used to encode the reset to default parameter:

<dev> default\_value when “true” indicates that default values shall be used.

### F.7.3.2 Picture Display Attributes <PDA>

The picture display attributes are defined by a set of parameters which describe some aspects of the display of a photographic source image at an application level (e.g. the placement of the photographic image).

Other aspects of the display which are strictly related to a given Videotex terminal are outside the scope of this annex and are left open to implementors. For example, rendering includes the alignment of the source image resolution to the physically displayed image or the alignment of the colour model of the source image to the colour capabilities of the physical display. Although such rendering is out of the scope of this annex, some guidelines are provided in Appendix II (informative).

In order to be independent of display hardware constraints, definitions in the following text are made with normalized co-ordinates. Normalized co-ordinates are derived from the unit screen concept, which defines co-ordinates as fractions of unity. The unit screen concept is illustrated in Figure F.5.

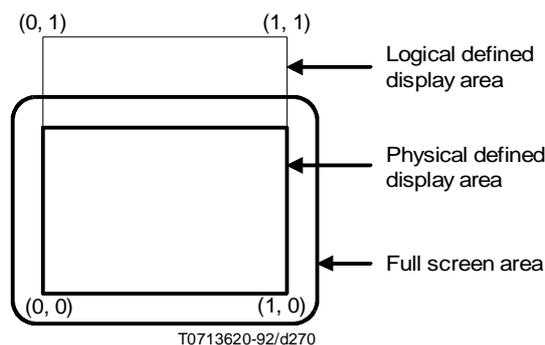


Figure F.5/T.101 – Unit screen

Figure F.5 shows the full screen area which is the part of the screen where photographic data can be displayed (e.g. full active lines and all active lines).

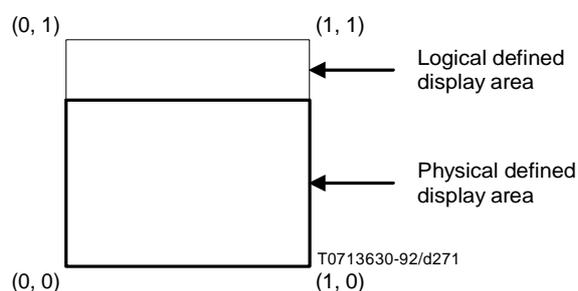
Inside the full screen area the physical Defined Display Area (physical DDA) is a rectangular area used for displaying images, text etc. Pictorial information within the physical DDA shall always be displayed, while pictorial information outside of the physical DDA, but inside the full screen area, may be displayed.

The Logical Defined Display Area (logical DDA) is a unit screen i.e. it is a square where the origin (0,0) is coincident with the bottom left corner of the physical DDA and whose one side is coincident with the longest side of the physical DDA.

The logical DDA may be partly outside the full screen area. Moreover, if it is possible to draw anywhere on the logical DDA, only the portion of the logical DDA coincident with the physical DDA shall be visible.

The virtual display space where the source image was encoded is referred to as the Source Defined Display Area (SDDA). This virtual space is mapped according to the value of the full screen display parameter to the full screen area or to the physical DDA.

The ratio of the width to the height of the SDDA is the source aspect ratio. One typical example, as shown in Figure F.6, is the commonly used 4/3 aspect ratio of some Videotex terminals.



**Figure F.6/T.101 – Source Defined Display Area (SDDA) with a 4/3 aspect ratio mapped to the physical DDA**

The picture display attributes are used to define a rectangular area in the logical DDA, this area is referred to as the photo-area. For photovideotex applications this area shall be filled with a photographic image. If the photographic image is larger than the photo-area, then the part of the image outside the photo-area shall not be displayed. However, when ITU-T Recommendation T.4 or CCITT Recommendation T.6 coding algorithms are used, local presentation facilities can be offered to display photographic images larger than the photo-area, some examples of local presentation facilities are given in Appendix VIII. If the photographic image is smaller than the photo-area, then part of the photo-area, shall be made transparent and therefore filled by the full background colour.

The following space allocation parameters are used to encode the photo-area:

- full screen display;
- source aspect ratio;
- photo-area location;
- photo-area size;
- picture placement;
- clear photo-area.

### F.7.3.2.1 Full Screen Display <FSD>

The full screen display parameter indicates that the SDDA shall be mapped to full screen area. All other parameters and sub-parameters are still functional. The full screen display parameter is defined using one sub-parameter:

<ful> full\_screen\_area display full screen.

If the value of the sub-parameter is “true” the SDDA shall be mapped to the full screen area.

NOTE – If this parameter is used, there is no guarantee for the physical display of the image outside the Physical Defined Display Area (PDDA) and there is no support for the alignment of the photographic plane to the alphamosaic and if applicable the alpha-geometric planes.

### F.7.3.2.2 Source Aspect Ratio <ASR>

The source aspect ratio parameter defines the ratio of the width over the height of the SDDA. This parameter is encoded using two integer values:

<araw> source\_width, the width of the source;

<arah> source\_height, the height of the source.

The source aspect ratio is obtained by dividing the value of <araw> by the value of <arah>. This is illustrated in Figure F.6.

NOTE – When the source is a facsimile scanner, it is likely that the source aspect ratio will be in relation with some paper format and not with a screen.

#### Example:

If the source is an A4 paper page scanned by a facsimile G3 apparatus, the ASR may then take the value  $215\text{mm}/297\text{mm} = 0.724$ .

### F.7.3.2.3 Photo-Area LOcation <LOC>

The photo-area location parameter defines the position of the bottom left hand corner of the photo-area in normalized co-ordinates relative to the origin of the logical DDA.

This parameter is encoded using two sub-parameters, as shown in Figure F.7, the horizontal and vertical normalized co-ordinates:

<loch> photo-area\_location\_horizontal, the normalized horizontal co-ordinate;

<locv> photo-area\_location\_vertical, the normalized vertical co-ordinate.

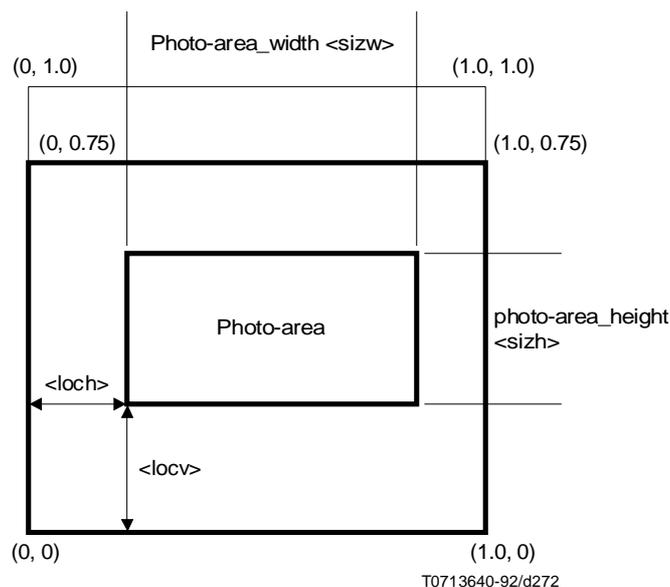


Figure F.7/T.101 – Unit screen coordinates and the photo-area

#### F.7.3.2.4 Photo-Area Size <PAS>

The photo-area size parameter defines the dimensions of the photo-area in normalized lengths. This parameter is encoded using two sub-parameters, the width and height of the photo-area:

- <size> photo-area\_width,      the normalized width of the photo-area;
- <size> photo-area\_height,      the normalized height of the photo-area.

This is illustrated in Figure F.7.

#### F.7.3.2.5 Picture PLacement <PPL>

The picture placement parameter defines how the source picture shall be positioned in the photo-area.

Using this parameter, the position of the photographic image relative to the photo-area is fixed. It is envisaged that this parameter can be used to ensure that the most important part of the photographic image is correctly positioned in the photo-area.

Two points are identified, one on the photographic source image and the other in the photo-area. These two points are defined to be coincident and hence the position of the photographic image in the photo-area is fixed (see Figure F.8).

This parameter is defined by two sets of sub-parameters:

- reference\_picture-point,      on the source image;
- offset\_picture-point,      in the photo-area.

In some situations the photo-area is not completely filled by the photographic image. In such cases it is assumed that this unfilled area is transparent.

In other situations the photographic image is larger than the photo-area. The part of the image outside the photo area shall not be displayed. However, when ITU-T Recommendation T.4 or CCITT Recommendation T.6 coding algorithms are used, local presentation facilities can be offered to display photographic images larger than the photo-area, some examples of local presentation facilities are given in Appendix VIII.

The following sub-parameters are used to encode the picture placement parameter:

- <refh> ref\_picture-point\_horiz,      the number of pixels from the picture origin to the reference\_ picture-point in the horizontal direction;
- <refv> ref\_picture-point\_vert,      the number of pixels from the picture origin to the reference\_ picture-point in the vertical direction;
- <offh> offset\_photo-area\_horiz,      the horizontal co-ordinate of the photo-area offset\_picture-point;
- <offv> offset\_photo-area\_vert,      the vertical co-ordinate of the photo-area offset\_picture-point.

The photo-area offset point and the picture reference point are coincident points.

In the default situation the picture origin shall be the default coincident point. This implies taking, for the sake of simplicity, the top left hand corner for both the photo-area and the photographic image.

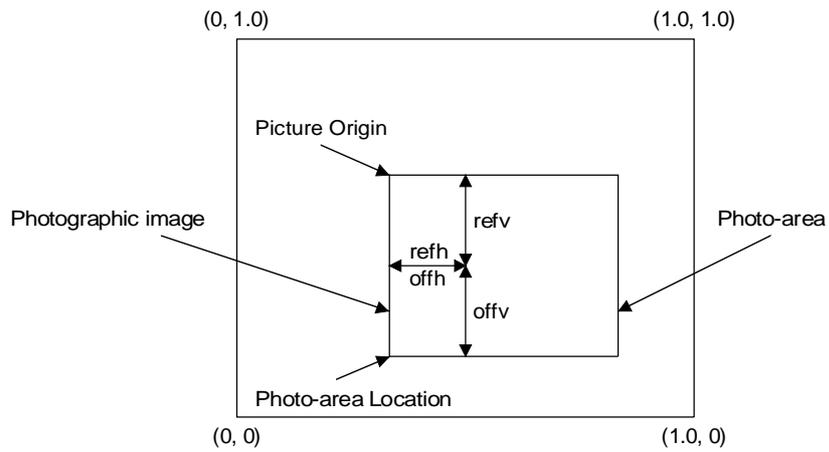
#### F.7.3.2.6 Clear Photo-Area <CPA>

A specified photo-area is cleared to transparent by the Clear Photo-Area parameter. The photo-area shall be specified using the parameters photo-area location and photo-area size. If the photo-area parameters location and size, are not specified, the default shall be the full DDA. The Clear Photo-Area parameter shall be encoded using one sub-parameter:

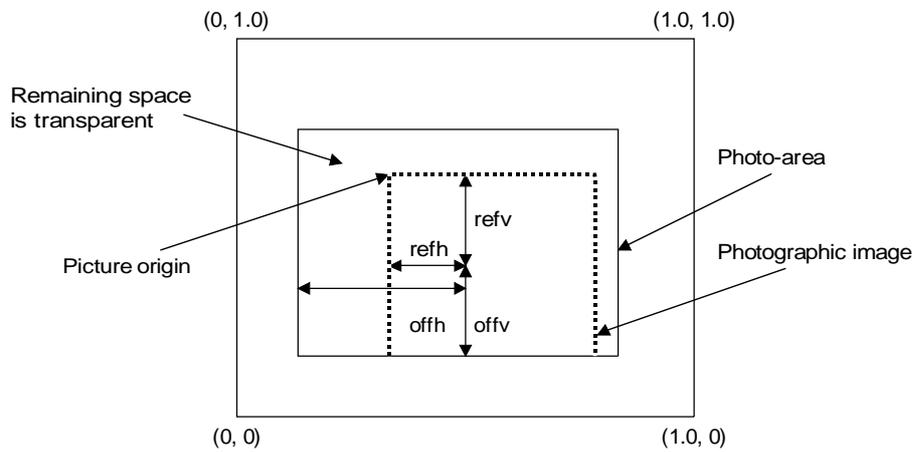
- <cle> clear\_photo\_area,      clear the photo-area.

If the value of the sub-parameter is “true” the photo-area shall be cleared.

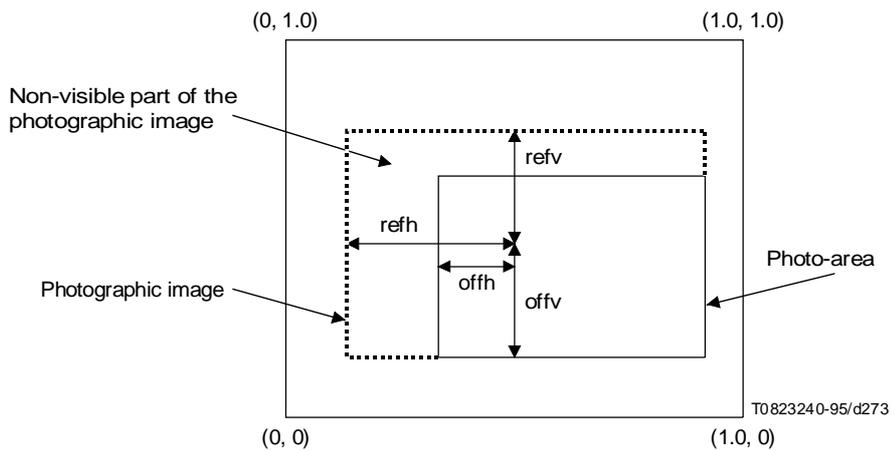
NOTE: – When the transparent colour does not exist in the terminal, the relevant background colour is used.



a) The photographic image and the photo-area are the same size



b) The photographic image is smaller than the photo-area



c) The photographic image is larger than the photo-area

Figure F.8/T.101

### F.7.3.3 Source Picture Attributes <SPA>

The Source Picture Attributes are defined by a set of parameters which describe the spatial composition of the photographic image. All of the parameters are related to the environment where the source image was encoded.

#### F.7.3.3.1 Source Picture Comments <PCT>

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

The source picture comment parameter indicates a text string as part of the photographic data introduced by the JPEG marker code COM (X'FFFE'). This text string is, however, not part of the alpha-mosaic mode. If this parameter is used the receiving terminal shall not be forced to (but may) display this text. Even if this text is displayed, any kind of "fall back" due to a limited character set is allowed (e.g. diacritical marks received are not displayed).

The source picture comments parameter includes character coded information about the source image, which normally does not form part of the Videotex frame. It may contain information about for example; the title of the source image, the author of the source, date of source production, copyright of source, size of the source, etc. Due to the simplicity and diversity of the possible uses of this option, no special logical record structuring of the text string is defined by this annex, it is left open for the application provider.

This parameter has two sub-parameters: `code_size <csz>` and `code_set_identifier <cid>`.

`Code_size <csz>` specifies in terms of the number of bits the length of each code. It can have the values 7, 8 or 16. If `<csz>` is 7 bits, bit 8 of a byte is not observed.

`Code_set_identifier <cid>` identifies the character set used (e.g. ITU-T Recommendation T.50 [14], ITU-T Recommendation T.51 [7], ITU-T Recommendation T.101 [5]) to code the text in the JPEG comment. (The default is ITU-T Recommendation T.51 [7]).

The <PCT> parameter has two sub-parameters:

`<csz><cid>`

where

<code>&lt;csz&gt;</code>	<code>code_size</code>	enumerated type (e.g. 7, 8, 16)
<code>&lt;cid&gt;</code>	<code>code_identifier</code>	enumerated type: <ul style="list-style-type: none"><li>– non-standard</li><li>– ITU-T Recommendation T.50 [14]</li><li>– ITU-T Recommendation T.51 [7]</li><li>– CCITT Recommendation T.61 [8]</li><li>– data syntax I</li><li>– data syntax II</li><li>– data syntax III</li></ul>

#### Example:

In a 7-bit environment for the transmission of the text it can be recommended to assign to the code positions 2/00 to 7/15 the character codes of ITU-T Recommendation T.51 [7] (or equivalent ISO 6937 [15]), which are the ITU-T and ISO recommended set of characters for the transmission of Latin characters (and which are a superset of the ITU-T Recommendation T.101 [1] primary and supplementary sets of graphic characters). All codes within the above code range should be recognized and accepted by the decoder, but their use (e.g. conversion into an internal code) is not within the scope of this annex.

For the transmission of control characters in ITU-T Recommendation T.51 [7] the code range 0/00 to 1/15 is reserved. The code assignment in the above range should follow the primary control set of ITU-T Recommendation T.101 [1]. All decoders supporting this function should recognise and accept characters of the above code range. It is recommended that as a minimum, with the exception of the necessary code extension characters of ISO 2022 [10] all possible characters of ITU-T Recommendation T.51 [7] should be recognized. The codes APR (0/13) (functionally equal to carriage return) and APD (0/10) (functionally equal to line feed) should be supported.

Thus:

<PCT><csz> “7-bit” <cid> “ITU-T Recommendation T.51”

(the exact coding of the sub-parameters is given in clause F.8)

An example of the use of this parameter is provided in Appendix IV (informative).

#### F.7.3.3.2 Source Picture Dimensions <PDS>

The dimensions of the source image are encoded with the source picture dimensions parameter. The dimensions of the image are given in terms of the number of pixels in the horizontal direction and the number of pixels in the vertical direction. The source picture dimensions parameter specifies the dimensions in terms of the highest resolution component. Two sub-parameters are therefore needed to encode this parameter:

<nph> num\_pixels\_horiz, the number of pixels in the horizontal direction;

<npv> num\_pixels\_vert, the number of pixels in the vertical direction.

The <npv> sub-parameter may be undefined with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding.

#### F.7.3.3.3 Source Pixel Density <PID>

The source pixel density parameter expresses for the general case the number of pixels per unit (e.g. inch or cm) of resolution, in both the vertical and the horizontal direction. For standardized formats this parameter shall make a direct reference to a standardized set, typically a CCIR Recommendation or ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18].

The source pixel density parameter is a key piece of information in the general field of image applications. The source pixel density information is necessary to ensure that the photographic data encoded in the source environment is processed and displayed to give the intended aspect ratio. Also the source photographic data can be converted to the resolution of the local display (or printing device), with a minimal introduction of artifacts (e.g. aliasing). However, it is worth noting that to ease the development of compatible applications, a few compatible families of Pixel Densities are recommended in the clause on profiles (clause F.11).

In the general case, the pixel density is expressed as the ratio of two integers, as for example 150/1. When the pixel density is expressed using decimal numbers, the proper rational value shall be found, for example 10.5 pixel/mm shall be represented by 21/2 pixel/mm. The parameter takes one of two forms:

<stf> standardized_form,	enumerated type:
	4:2:2 625 lines
	4:2:2 525 lines
	2:1:1 625 lines
	2:1:1 525 lines
	CIF
	Not applicable
	8 × 3.85 (hor × vert) pixels/mm
	8 × 7.7 pixels/mm
	8 × 15.4 pixels/mm
	16 × 15.4 pixels/mm
	200 × 200 pixels/25.4 mm
	240 × 240 pixels/25.4 mm
	300 × 300 pixels/25.4 mm
	400 × 400 pixels/25.4 mm

or

<p_h_num> pixels_per_unit_horizontal_numerator,	integer;
<p_h_den> pixels_per_unit_horizontal_denominator,	integer;
<p_v_num> pixels_per_unit_vertical_numerator,	integer;
<p_v_den> pixels_per_unit_vertical_denominator,	integer;
<unt> unit,	enumerated (e.g. pixels/cm, pixels/inch, pixels/mm).

For a multi-component picture, the <PID> specification shall be made in sequence for each component.

**Example:**

<PID> (p\_h\_num<sub>1</sub>, p\_h\_den<sub>1</sub>, p\_v\_num<sub>1</sub>, p\_v\_den<sub>1</sub>, unt)  
 <PID> (p\_h\_num<sub>2</sub>, p\_h\_den<sub>2</sub>, p\_v\_num<sub>2</sub>, p\_v\_den<sub>2</sub>, unt)  
 <PID> (150/1, 150/1, inch)  
 <PID> (75/1, 75/1, inch)  
 <PID> (75/1, 75/1, inch)

The Source Component Description parameter defines the order of the components.

NOTE 1 – The parameters Source Picture Dimensions and Source Picture Density describe the source image. They are included to provide the terminal with the information which may be required for scaling or clipping of an image. If the source image is derived from a bit map, only the Source Picture Dimensions parameter contains information which is useful to the terminal. The value of the “<stf> standardized form” in this case is set to not applicable.

ITU-T Recommendation T.30 [20] indicates that some terminal may interpret resolution  $8 \times 7.7$  and  $16 \times 15.4$  ppm as  $200 \times 200$  and  $400 \times 400$  ppi resolutions respectively. Terminal complying with this annex may also take this approximation into account.

NOTE 2 – Recommendation T.4 [17] also defines the option 1728 pixels along 151 mm, i.e. 11,44 pixels/mm for A5 or A6 scanner. The use of this horizontal definition is left for further study.

NOTE 3: – Resolution  $240 \times 240$  is defined by ITU-T Recommendation T.563 [21] and reported in this annex. It is, however, not recommended to have it used when scanning the images.

**F.7.3.3.4 Source SWEEP Direction <SWD>**

The Source Sweep Direction parameter specifies the direction in which the image is built up. There are two possible values of the sweep direction of a row:

top to bottom, bottom to top.

There are two possible values of the sweep direction of a line:

left to right, right to left.

The sweep direction is completely specified using two sub-parameters.

<sdir> the sweep direction of a row;  
 <sdil> the sweep direction of a line.

**F.7.3.3.5 DC Images <DCI>**

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

Small images  $1/64^{\text{th}}$  of the size of the whole image can be obtained by decoding and extracting the DC values. A small image can also be coded as a normal JPEG image with only the DC values used. For a description of the term, DC value, reference should be made to Appendix I (informative).

The DC Image parameter is encoded with one sub-parameter:

<dcv> dc\_value, an indication of whether DC values only are in use.

This parameter can have two values: “true” or “false”. When the value is set to “true”, decoding shall take place to display DC sized images. When the value is set to “false”, decoding shall take place to produce a full size image.

**F.7.3.4 Source Signal Attributes <SSA>**

The Source Signal Attributes describe the components which form the image and the relationship between the components. The parameters used to define the Source Signal Attributes allow for the use of a large variety of colour models, interleave structures, etc.

The Source Signal Attributes <SSA> doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding.

**F.7.3.4.1 Source Component Description <SCD>**

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

An image can be formed from one or more source components. A monochrome image requires one component, a colour image needs at least three. The different sets of components which can be used are each assigned a separate code. The number of components used is defined by the set. The order of the components which is required by some other parameters is defined by this parameter.

Another important source picture characteristic is whether the image is block or component interleaved. The definition of that characteristic shall be given in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] and not in this annex.

For example, if the set is RGB (Red, Green, Blue) then the number of components is three. The component sets which can be selected are:

- R, G, B;
- Y, C<sub>B</sub>, C<sub>R</sub>;
- C, M, Y, K;
- Y.

The sub-parameter

<com> component

is used to send the enumerated value identifying the component set.

#### **F.7.3.4.2 Source Component Data Precision <CDP>**

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

The Source Component Data Precision parameter specifies, for each image component, how many bits are required to represent each component sample associated with a pixel. For example, a luminance component with 256 grey levels needs to be represented using 8 bits.

The sub-parameter expressing the precision in terms of bits/component is

<cpt> component\_data\_precision

For a multi-component picture, the sub-parameter <cpt> shall be sent for each component in the same <CDP>.

#### **Example:**

For a three components image with 8 bits per component :

<CDP> (cpt1) (cpt2) (cpt3)

<CDP> (8) (8) (8)

#### **F.7.3.4.3 Source Component Order <CMO>**

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

The Source Component Order parameter specifies the order in which the components are sent to the terminal. The order is specified in terms of the order defined by the source component description parameter.

For example if the parameter source component description has the value "RGB" then the order is defined as red 1<sup>st</sup>, green 2<sup>nd</sup>, blue 3<sup>rd</sup>. If, however, the components are sent with the order green, blue, red the value of source component order shall be 2,3,1. This parameter only defines the order of component transmission, for all other parameters (i.e. <PID>, <CDP>, <LAS>) concerned with the order of the components, refer to the source component description parameter.

The sub-parameter

<cor> component\_order

shall be sent for each component in the same <CMO>

#### **Example:**

For the above example, the <CMO> will be :

<CMO> (cor1)(cor2)(cor3)

<CMO> (2)(3)(1)

**F.7.3.4.4 Source Level Assignment <LAS>**

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

One explicit example of the use of this parameter is provided by the CCIR Recommendation 601, Part 1 [9]. In this CCIR Recommendation, reference levels are used to fix the values (within the range 0 to 255) used to define certain colours of the source. For example, for a luminance signal the values used for black (16) and the value used for white (235) are fixed.

The Source Level Assignment parameter can be assigned a value which is either an enumerated type or an integer type.

The only enumerated type which is defined indicates that the level assignment follows the CCIR Recommendation 601, Part 1 [9].

If the level assignment is not done according to CCIR Recommendation 601 [9], integer values shall specify the lowest and highest level per component. The full range of the colour component is defined by the source component data precision parameter.

The source level assignment parameter is encoded with one of two sets of sub-parameters:

<fix> fixed\_assignment      an enumerated value refers to a standardized level assignment

or

<low> lowest\_level            lowest level of the component

<hi> highest\_level            highest level of the component

**F.7.3.5 Source Coding Algorithm Attributes <SCA>**

The source coding algorithm attributes identify two basic aspects:

- 1) the technique being used to compress and encode the pixel data; and
- 2) the application oriented coding functions.

There are a large number of techniques available, however only those techniques which have been standardized shall be included here.

The application oriented coding function is the term used to describe a few functions left open by a given coding technique for handshaking with the application.

Only techniques developed by ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] and ITU-T Recommendation T.4 [17] and CCITT Recommendation T.6 [18] and some simple extensions shall be considered. All the parameters described here refer to these techniques.

Appendix I (informative) contains a description of the compression techniques which have been developed by ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] and should be consulted in order to fully understand the meaning of these parameters.

Not all combinations of the parameters are possible. Table F.2 summarizes all the permitted combinations, for ITU-T Rec. T.81 | ISO/IEC 10918-1 [13].

**Table F.2/T.101 – JPEG coding mode**

Hierarchical Mode	No						Yes					
Type of Algorithm	DCT				DPCM spatial		DCT				DPCM spatial	
Built up Mode	Seq.		Prog.		Seq.		Seq.		Prog.		Seq.	
Entropy Coding Technique	H	A	H	A	H	A	H	A	H	A	H	A
Seq. Sequential image build up Prog. Progressive image build up H Huffman coding A Arithmetic coding NOTE – The leftmost column corresponds to the modes of operation of the minimum capability requested for all DCT based systems referred to as the BASELINE system.												

### **F.7.3.5.1 JPEG Coding Mode <JPG>**

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

The JPEG coding mode parameter sets in an abbreviated form the basic modes of operation of the JPEG coding technique. This is typically specified in the ITU-T Rec. T.81 | ISO/IEC 10918-1 [13], but via a marker code structure and in an expanded form. The intention is to give at the application level, before decoding the image, a compact description of the JPEG coding modes which have been used to compress the images conveyed by the photographic data syntax. This shall be in accordance with the profile definitions.

In the following, all the JPEG modes are described. However, in the photovideotex profiles (see clause F.11) only a subset is used.

#### **Hierarchical mode**

Hierarchical mode allows the resolution of the image to be increased in stages. From the user's point of view, first a low resolution image is completely displayed (either sequentially or progressively). After the completion of the first image a second iteration increases the resolution of the image, such iterations may be repeated several times.

The increase in resolution can be an increase of spatial resolution by a factor of two in either the horizontal or the vertical direction (or both). It can also be a refinement of image quality at a given spatial resolution.

The possible values are:

- hierarchical;
- non-hierarchical.

#### **Type of algorithm**

Two types of compression algorithm are used in the ITU-T Rec. T.81 | ISO/IEC 10918-1 [13]. One is based on the Discrete Cosine Transform (DCT) which is lossy. The other is a specialized lossless algorithm based on a spatial prediction technique (DPCM).

The possible values are:

- DCT;
- DPCM.

#### **Build-up mode**

An image can be reproduced on the screen of the user's terminal using either a sequential or a progressive build-up mode.

In the sequential mode each image component is built up to full precision in one pass – one scan per component in non-interleaved mode (e.g. Y, C<sub>B</sub>, C<sub>R</sub>, coded in separated scan) or one scan per group of components in interleave mode (e.g. Y in one scan, and C<sub>B</sub>, C<sub>R</sub>, interleaved in one scan). If the algorithm type is DPCM the image is built up pixel by pixel, if the algorithm is of DCT type the image is built up from pixel blocks (a pixel block being 8 × 8 pixels).

In the progressive mode the entire image is built up from a number of scans, each successive scan improving the entire image quality. The build-up of a scan can occur in pixel blocks or on a pixel by pixel basis according to the type of algorithm, as for the sequential mode.

In both cases the order of colour build up is defined by the source component order parameter.

Two values are possible:

- sequential;
- progressive.

## Entropy coding technique

One of two entropy coding techniques can be applied to the processed data in order to achieve further compression.

These two techniques are:

- Huffman Coding;
- Arithmetic Coding.

A single sub-parameter <cmp> is used to encode the values assigned to the parameter <JPG>. The use of this sub-parameter shall be defined in clause F.8.

### F.7.3.5.2 Encoding Table Management <ETM>

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

Depending on the type of compression technique which has been selected, different tables are needed to decode the image. The JPEG compression algorithm provides the tools for the transmission of the table data and for the attachment of the tables to the components of the image. However, the "JPEG algorithm" leaves to the application the task of properly managing all tables. Typically, at the application level a library of entropy tables (Huffman) and/or quantization tables may be maintained and managed. To achieve this functionality the photographic data syntax contains four , sub-parameters:

- Table\_type;
- Table\_id;
- Table\_status; and
- Scaling\_factor.

It is important to note that the number of tables in use at one time (in a coder or decoder) is constrained as described in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13]. At the application level the location of the tables, i.e. whether they are the storage area of the "in use environment" or in the storage area of the application, is managed. The tables are transferred as part of the JPEG data stream. Considering that the JPEG data stream and the <ETM> parameters are sent separately, a rule is needed to associate an occurrence of an <ETM> parameter with a particular table. The relation between the two is simply time-dependent, this means that the first <ETM> parameter sent shall correspond to the first table transferred in the JPEG data stream, the second to the second, etc.

For specific applications using the same set of custom tables for all subsequent pictures, it shall be ensured that the downloading of the tables has been successful.

#### Table\_type <ttp>

Three types of table are used:

- quantization tables;
- Huffman tables; and
- conditioning tables.

Quantization tables are needed for the lossy compression technique. Huffman tables are needed when Huffman coding has been selected for the entropy coding technique. When Table\_type is set to Huffman it shall refer to the set of tables, the AC Huffman table and the DC Huffman table of a particular image component. Conditioning tables are needed when arithmetic entropy coding has been selected for the entropy coding technique.

Two values are possible for the sub-parameter <ttp>:

- Quantization;
- Huffman.

NOTE – For arithmetic condition tables no valid subparameter exists since the Start Of Image (SOI) marker would reset the arithmetic tables to default.

The Quantization tables contain two tables:

- Luminance quantization table;
- Chrominance quantization table.

The Huffman tables contain the following tables:

- Huffman table for luminance DC differences;
- Huffman table for chrominance AC differences;
- Huffman table for DC luminance coefficients;
- Huffman table for AC chrominance coefficients.

#### **Table\_id <tid>**

The Table\_id sub-parameter is needed to uniquely identify the tables to be used within a given application. In the case of Huffman tables, a single value of Table\_id shall identify both DC and AC tables of a given image component. The number of tables of each type is theoretically not limited at the application level, but is limited for practical reasons in the terminal implementation. The number of tables of each type, used at any one time, is referred to as the “in-use environment” and is directly related to the profile definitions (clause F.11). The coding of this sub-parameter provides a mechanism for assigning the tables to a specific area of the terminal’s “in-use environment”. This sub-parameter shall take the form of an integer.

The sub-parameter

<tid> identifier of the table

shall be used to uniquely identify the table.

#### **Table\_status <tst>**

The table\_status shall be assigned the following values:

- load default table;
- use the current table;

or

- table to be transferred.

The value “load default table” indicates that the decoder shall load, in the current tables, the default tables which are permanently stored in the terminal.

The value “use the current table” indicates that the decoder shall use the tables which are currently in use in the current tables. These could be default tables or the tables last sent.

The value “table will be transferred” indicates that new tables shall be transferred as part of the transparent data and shall be used as current tables.

The sub-parameter

<tst> status of the table

shall be used to reflect the status given by the application to the table. The following three values are possible:

- load table default;
- use the current table;
- table will be transferred.

#### **Scaling\_factor <sfr>**

This sub-parameter may be used to control the image compression ratio using the default Quantization tables. The scaling factor parameter is applied to all the values of the default quantization tables, thus, allowing to control the quality of the displayed picture and avoiding the transmission of the Quantization tables. The scaling factor parameter, which is a real number, is represented by two sub-parameters <sfr\_num> and <sfr\_den> which represent respectively:

- the numerator of the scaling factor ;
- the denominator of the scaling factor.

### **F.7.3.5.3 Application Marker codes Assignment <AMA>**

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

The JPEG algorithm has reserved a marker code space for signalling application-dependent functionalities, not already provided by the application syntax. This gives the application some flexibility to adapt and to extend the functionalities defined in this photographic data syntax to not fully specified or as yet undefined applications.

As one example, the use of the still picture JPEG algorithm for animated images (e.g. one image per second) – not to be confused with motion images – is considered as a possible straight forward extension of this photographic syntax. In that case, one application marker code is reserved to signal this extension of the photographic data syntax and to convey the specific additional data required by such an application outside the Videotex service.

Another example, of an option within the Videotex service, is to allow for a special terminal display facility. A colour palette optimized to a given photographic image is embedded into the photographic data, with a specific marker code.

In these particular cases, the JPEG application marker codes shall be used to signal extension of the photographic syntax and to convey the specific additional data required. The formatting of this data is either fully application dependent or is specified in the profiles clause (clause F.11).

The sub-parameter shall take the form of an enumeration

<mak> application marker code assignment

the value shall be in the range imposed by the JPEG codes “reserved for application segments (APP<sub>n</sub>)” : X 'FFE0' – X 'FFEF'

In this annex, the following application marker codes shall be assigned:

The marker code X'FFE0' shall be assigned to the “animated images application”.

The marker code X'FFE1' shall be assigned to the “colour palette definition”.

By default and on reception of <RTD> command, there is no application marker.

### **F.7.3.5.4 T4 Coding Mode <T4C>**

This parameter doesn't apply with ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] coding. It shall not be taken into account if received.

The T4 coding mode parameter sets in an abbreviated form the basic modes of operation of the T4 coding technique as specified in CCITT Recommendation T.4 [17].

This parameter is exclusive with the T6 coding mode <T6C> and shall be used when the PI (Picture Identifier) is set to T4.

The possible values are:

- No compression algorithm used;
- Monodimensional compression algorithm;
- Bidimensional compression algorithm with K = 2;
- Bidimensional compression algorithm with K = 4.

### **F.7.3.5.5 T6 Coding Mode <T6C>**

This parameter does not apply with ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] coding. It shall not be taken into account if received.

The T6 coding mode parameter sets in an abbreviated form the basic modes of operation of the T6 coding technique as specified in CCITT Recommendation T.6 [18].

This parameter is exclusive with the T4 coding mode <T4C> and shall be used when the PI (Picture Identifier) is set to T6.

The possible values are:

- No compression algorithm used;
- Bidimensional compression algorithm with K = infinite.

### F.7.3.6 Transmission Channel Attributes <TCA>

The Transmission Channel Attributes describe the transmission characteristics of the channel used to transport the photographic data.

#### F.7.3.6.1 Translation Mode Encoding <TME>

The Translation Mode Encoding parameter indicates which of the translation modes shall be used to transport the photographic data. The translation mode values shall be allocated as shown in Table F.4 (see F.9.2). In order to offer compatibility with 7-bit systems, a mode supporting 7-bit transparency shall be allowed. Due to the inefficiency of 7-bit modes their use is not recommended.

The sub-parameter

<mod> mode\_of\_translation

indicates which translation mode shall be used to send the photographic data.

## F.8 Coding rules

### F.8.1 Purpose

Coding rules shall be applied to the coding of the photographic header. A uniform structure is achieved by the use of attributes and parameters. A-7 bit coding technique shall be used to facilitate the implementation of the photographic data syntax in a 7- or an 8-bit environment. In general, a byte will be used to encode the 7-bit codes, bit 8 is, however, never used for coding purposes.

### F.8.2 General rules for coding the header

NOTE 1 – In order to allow for future possible extensions of the attributes and parameters, a full set of coding is described in this subclause. The current coding of attributes and parameters does not utilize all the possibilities given in this subclause.

The method of coding the various parameters of the various attributes consists of one 7-bit code for the attribute followed by one 7-bit code for the parameter. The attributes and the parameters shall be followed by a number of sub-parameters. A sub-parameter code shall be represented using three fields: type, length and value. The sub-parameter field type, shall be encoded using one 7-bit code. The sub-parameter field length consists of a number of 7-bit codes containing the length (number) of 7-bit codes used by the sub-parameter field value. The sub-parameter field value consists of a number of 7-bit codes.

The abbreviations defined below are used in subsequent examples to illustrate how attributes, parameters and sub-parameters are coded. Figure F.9 shows the structure of a sub-parameter code.

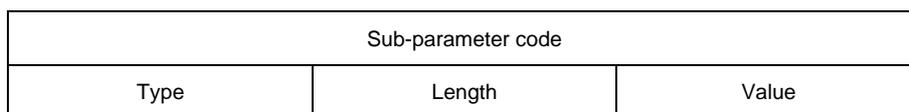


Figure F.9/T.101 – Structure of a sub-parameter code

**Example 1: Attribute, parameter and sub-parameter**

Attribute code (AC)

Parameter code (PC)

Sub-parameter code (SC)

$SC = ST_x + SL_x + SV_x$

Sub-parameter field Type ( $ST_x$ )

Sub-parameter field Length ( $SL_x$ )

Sub-parameter field Value ( $SV_x$ )

The following items a) to u) contain the coding rules which shall be observed when coding the attributes, parameters and sub-parameters of the photographic header:

- a) in a 7-bit environment bit 8 of a byte shall be set to zero, only 7 of the 8 bits are used for coding purposes;
- b) an attribute code shall be followed by a parameter code;
- c) a parameter code shall be followed by at least one sub-parameter code;

**Example 2:**

[AC PC SC1]

**Example 3:**

[AC PC (ST1 + SL1 + SV1)]

- d) if a sub-parameter code or group of sub-parameter codes is sent more than once, each occurrence of the code or group of codes shall be preceded by the relevant parameter code;

**Example 4:**

[AC PC SC1 SC2 SC<sub>x</sub>]

- e) “|” means that the sender shall make a choice which sub-parameter or group of sub-parameters shall be sent. In the case of a group of sub-parameters all members of the group of sub-parameters shall always be sent;
- f) a new attribute code shall follow directly after the last sub-parameter code of the previous attribute;

**Example 5:**

[AC PC SC1 SC2] + [AC PC (ST1 + SL1 + SV1)] + ...

- g) all attributes shall be coded by one byte from column 2 of Figure F.10;
- h) all parameters shall be coded by one byte from column 3 of Figure F.10;
- j) a sub-parameter consists of a **type field**, a **length field** and a **value field**;
- k) the **sub-parameter field type** shall be coded by one byte from column 4 of Figure F.10;
- l) the **sub-parameter field length** shall be coded in the following way:

The length indicating the number of subsequent bytes shall have the following structure, where:

bit 8 = not used;

bit 7 = extension flag;

bits 6 to 1 = specify the number of bytes used to encode the sub-parameter value.

Bit 7 of each byte shall be the extension flag. The value shall be specified in binary notation as an unsigned integer using bits 6 to 1 with the weights  $2^5$ ,  $2^4$ ,  $2^3$ ,  $2^2$ ,  $2^1$  and  $2^0$ , respectively. If the value is less than, or equal to 63, it shall be represented by one byte and the extension flag shall be set to ZERO.

If the value is larger than 63, it shall be represented by more than one byte. The most significant part of this value shall be contained in the byte recorded first. The extension flag shall be set to ONE in all bytes except the last, where it shall be set to ZERO.

- m) the sub-parameter field value shall be encoded using a number of bytes, the number shall be indicated in the sub-parameter field length;
- n) the following sub-parameter types shall be defined: integer, real, normalized, decimal, enumeration, Boolean and string;
- o) encoding of an integer value:
  - the byte or bytes of the **sub-parameter field value** shall be equal to an integer value, consisting of bits 6 to 1 of the first byte, followed by bits 7 to 1 of the second byte, followed by bits 7 to 1 of each byte in turn up to and including the last byte of the **sub-parameter field value**;
  - the value of the integer binary number shall be derived by numbering the bits of the bytes, N representing this number, in the **sub-parameter field value**, starting with bit 1 of the last byte as bit one ( $N = 1$ ) and ending the numbering with bit 6 of the first byte;
  - each bit is assigned a numerical value  $2^{N-1}$ , where N is its position in the above numbering sequence. The integer value shall be obtained by summing the numerical values assigned to each bit for those bits which are set to one, excluding bit 7 of the first byte;
  - negative values are represented by the two's complement notation, which uses bit 7 of the first byte as a sign bit (0/1 positive/negative values).

According to the above rules the integer values +6837 and –6837 shall be coded:

**Example 6:**

Coded representation of a positive integer number +6837

Byte 1 is X0110101

	X	Sign	N = 13	N = 12	N = 11	N = 10	N = 9	N = 8
	X	0	1	1	0	1	0	1
$2^{N-1}$	X	+	4096	2048	0	512	0	128

Byte 2 is X0110101

	X	N = 7	N = 6	N = 5	N = 4	N = 3	N = 2	N = 1
	X	0	1	1	0	1	0	1
$2^{N-1}$	X	0	32	16	0	4	0	1

$+6837 = 4096 + 2048 + 512 + 128 + 32 + 16 + 4 + 1$

**Example 7:**

Coded representation of a negative integer number –6837

Byte 1                      Byte 2  
 X 1 0 0 1 0 1 0      X 1 0 0 1 0 1 1      X = bit 8 and is not used

- p) encoding of a real value;

NOTE 2 – ITU-T Recommendation X.209 [“Specification of basic encoding rules for Abstract Syntax Notation One (ASN1)”] can be consulted for further clarification on the coding of a real value. The coding of a real value described below is based on ITU-T Recommendation X.209.

A real value shall be specified by the following formula involving three integers [encoded according to item o)], M, B and E:

$$\text{value} = M \times B^E$$

M is the mantissa, B the base and E the exponent. M and E may take any values, positive or negative, while B shall take only the value 2, 8 or 16. All combinations of M, B and E are permitted.

The mantissa M shall be represented by a sign S, a non-negative integer value N and a binary scaling factor F, such that:

$$M = S \times N \times 2^F, 0 \leq F < 4, S = +1 \text{ or } -1$$

NOTE 3 – The freedom to choose F is provided to enable easier generation of the transfer format by eliminating the need to align the implied decimal point of the mantissa with an octet boundary. The existence of F does not noticeably complicate the task of the decoder. F is only needed with base 8 and 16 representations.

The coding of S, B, F, E, and N shall be done in the following way:

- 1) bit 8 of the first byte is not used;
- 2) **S** = bit 7 of the first byte shall be 1 if S is –1 and 0 otherwise;
- 3) **B** = bits 6 to 5 of the first byte shall encode the value of the base as shown in Table F.3;
- 4) **F** = bits 4 to 3 of the first byte shall encode the binary scaling factor F as an unsigned integer;
- 5) **E** = bits 2 to 1 of the first byte shall encode the format of the exponent as follows:
  - if bits 2 to 1 are 00, then the second byte encodes the value of the exponent as a one byte integer value;
  - if bits 2 to 1 are 01, the second and third bytes encode the value of the exponent as a two byte integer value;
  - if bits 2 to 1 are 10, then the second, third and fourth bytes encode the value of the exponent as a three byte integer value;
  - if bits 2 to 1 are 11, then the second byte encodes the number of bytes, say X, (as a positive integer value) used to encode the value of the exponent. The third byte up to including the (X + 3)<sup>th</sup> byte contains the value of the exponent as an integer value.
- 6) **N** = the remaining bytes encode the value of the integer N as a positive integer number.

**Table F.3/T.101 – Base Allocation for real numbers**

Bits 6 to 5	Base
00	Base 2
01	Base 8
10	Base 16

**Example 8:**

Coding of the positive real number +0.58984375

Byte 1 X0000000 Real number header

X	Sign	B	B	F	F	E	E
X	0	0	0	0	0	0	0
X	0	Base 2		F = 0		Byte 2 = E	

Byte 2 X1110011 Exponent (two's compliment)

	X	Sign	N = 6	N = 5	N = 4	N = 3	N = 2	N = 1
	X	1	1	1	0	0	1	1
$2^{N-1}$	X	-	32	16	0	0	2	1

$$E = (32 + 16 + 2 + 1) - 64 = -13$$

Byte 3 X0100101 Mantissa (MSB)

	X	Sign	N = 13	N = 12	N = 11	N = 10	N = 9	N = 8
	X	0	1	0	0	1	0	1
$2^{N-1}$	X	+	4096	0	0	512	0	128

Byte 4 X1100000 Mantissa (LSB)

	X	N = 7	N = 6	N = 5	N = 4	N = 3	N = 2	N = 1
	X	1	1	0	0	0	0	0
$2^{N-1}$	X	64	32	0	0	0	0	0

$$N = 4096 + 512 + 128 + 64 + 32 = 4832$$

$$M = S \times N \times 2^F = + 4832$$

$$\text{Value} = M \times B^E = 4832 \times 2^{-13} = 0.58984375$$

NOTE 4 – F is set to zero because the base is 2. E is selected to give an integer value for N. The same real number can be coded in different ways.

**Example 9:**

Coding of a negative real number -0.58984375

Byte 1 X 1 0 0 0 0 0 0 real number header

Byte 2 X 1 1 1 0 0 1 1 exponent (two's compliment)

Byte 3 X 0 1 0 0 1 0 1 mantissa (MSB)

Byte 4 X 1 1 0 0 0 0 0 mantissa (LSB)

$$S = -1, N = 4832, F = 0, B = 2, E = -13$$

NOTE 5 – In this annex, the term “normalized” is not used in its classical sense because it includes +1 or -1.

- q) encoding of a normalized value:
- the range of normalized values shall be from  $-1$  to  $+1$ , inclusive;
  - if bit 6 of the first byte is set to one, all other bits shall be set to zero with the exception of the sign bit, bit 7;
  - the bytes of the **sub-parameter field value** shall be equal to a normalized value, consisting of bits 6 to 1 of the first byte, followed by bits 7 to 1 of the second byte, followed by bits 7 to 1 of each byte in turn up to and including the last byte of the sub-parameter field value;
  - the value of the normalized binary number shall be derived by numbering the bits of the bytes in the **sub-parameter field value**, starting with bit 6 of the first byte as bit one ( $N = 1$ ) and ending the numbering with bit 1 of the last byte;
  - each bit shall be assigned a numerical value  $1/(2^{N-1})$ , where  $N$  is its position in the above numbering sequence. The normalized value is obtained by summing the numerical fractions assigned to each bit for those bits which are set to one, excluding bit 7 of the first byte;
  - negative values shall be represented by the two's complement, which uses bit 7 of the first byte as a sign bit (0/1 positive/negative values).

**Example 10:**

Coding of a positive normalized number  $+0.58984375$

Byte 1	Byte 2	
X 0 0 1 0 0 1 0	X 1 1 1 0 0 0 0	X = not used

**Example 11:**

Coding of a negative normalized number  $-0.58984375$

Byte 1	Byte 2
X 1 0 0 1 1 0 1	X 0 0 1 0 0 0 0

- r) encoding of a decimal value:
- the bytes of the **sub-parameter field value** shall be the codes 3/0 to 3/9 representing the values 0 to 9, 3/10 representing a decimal point, 3/11 representing a positive number and 3/12 representing a negative number. The sign byte should be first.

**Example 12:**

Coding of decimal number  $-75,2$

Byte 1	Byte 2	Byte 3
X 0 1 1 1 1 0 0	X 0 1 1 0 1 1 1	X 0 1 1 0 1 0 1
3/12 = negative	3/7 = 7	3/5 = 5

Byte 4	Byte 5
X 0 1 1 1 0 1 0	X 0 1 1 0 0 1 0
3/10 = ,	3/2 = 2
X = not used	

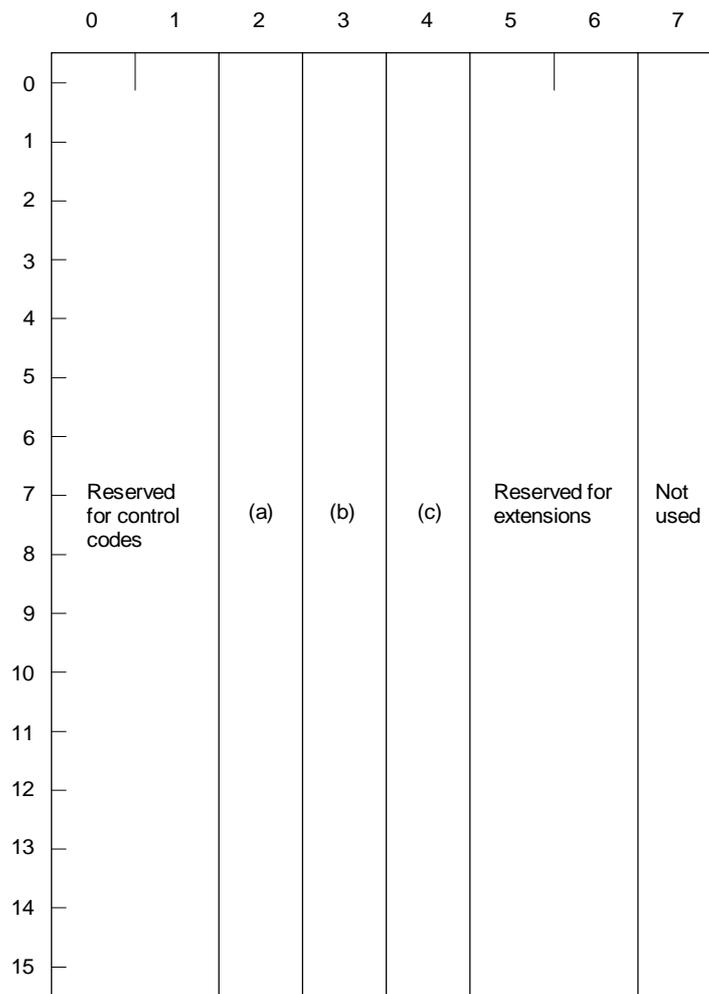
- s) encoding of an enumerated value:
- an enumerated value shall be encoded with an integer. The value of the integer indicates which value is assigned to a sub-parameter;
- t) encoding of a Boolean value:
- if the Boolean value is “FALSE”, the value of the byte shall be zero;
  - if the Boolean value is “TRUE”, the value of the byte shall be any non-zero number (bit 8 is not used);
- u) encoding of a string value:
- the bytes of the **sub-parameter field value** shall be the codes of the 7-bit environment of ITU-T Recommendation T.51 [7].

**Example 13:**

Coding of a string "ETSI"

Byte 1	Byte 2	Byte 3	Byte 4
X 1 0 0 0 1 0 1	X 1 0 1 0 1 0 0	X 1 0 1 0 0 1 1	X 1 0 0 1 0 0 1
E	T	S	I

X = not used



T0817290-94/d274

- (a) Photographic attributes
- (b) Photographic parameters
- (c) Sub-parameter types

**Figure F.10/T.101 – Photovideotex code allocation**

### F.8.3 Photographic header code assignment

#### F.8.3.1 Attribute codes

Parameter Status Attribute	<PSA>	2/0
Picture Display Attributes	<PDA>	2/1
Source Picture Attributes	<SPA>	2/2
Source Signal Attributes	<SSA>	2/3
Source Coding Algorithm Attributes	<SCA>	2/4
Transmission Channel Attributes	<TCA>	2/5

The Source Signal Attribute <SSA> 2/3 is not used with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

#### F.8.3.2 Parameter codes

##### F.8.3.2.1 Parameter Status Attribute

Reset to Default	<RTD>	2/0 3/0
------------------	-------	---------

##### F.8.3.2.2 Picture display attributes

Full Screen Display	<FSD>	2/1 3/0
Source Aspect Ratio	<ASR>	2/1 3/1
Photo-area Location	<LOC>	2/1 3/2
Photo-area Size	<PAS>	2/1 3/3
Picture Placement	<PPL>	2/1 3/4
Clear Photo-area	<CPA>	2/1 3/5

##### F.8.3.2.3 Source picture attributes

Source Picture Comments	<PCT>	2/2 3/0
Source Picture Dimensions	<PDS>	2/2 3/1
Source Pixel Density	<PID>	2/2 3/2
Source Sweep Direction	<SWD>	2/2 3/3
DC Images	<DCI>	2/2 3/4

The Source Picture Comments parameter <PCT> 2/2 3/0 doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

##### F.8.3.2.4 Source signal attributes

All these parameters do not apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. They shall not be taken into account if received.

Source Component Description	<SCD>	2/3 3/0
Source Component Data Precision	<CDP>	2/3 3/1
Source Component Order	<CMO>	2/3 3/2
Source Level Assignment	<LAS>	2/3 3/3

##### F.8.3.2.5 Source coding algorithm attributes

JPEG Coding Mode	<JPG>	2/4 3/0
Encoding Table Management	<ETM>	2/4 3/1
Application Marker Codes Assignment	<AMA>	2/4 3/2
T4 Coding Mode	<T4C>	2/4 3/3
T6 Coding Mode	<T6C>	2/4 3/4

##### F.8.3.2.6 Transmission channel attributes

Translation Mode Encoding	<TME>	2/5 3/0
---------------------------	-------	---------

### F.8.3.3 Sub-parameter codes

Integer	4/0
Real	4/1
Normalized	4/2
Decimal	4/3
Enumeration	4/4
Boolean	4/5
String	4/6

## F.8.4 Encoding of photographic header parameters

### F.8.4.1 Parameter Status Attribute <PSA>

#### F.8.4.1.1 Reset To Default <RTD>

<RTD> := 2/0 3/0 <dev>

<dev> := 4/5 <length> <value> (Boolean)  
<value> := TRUE;            default values  
<value> := FALSE;         not default values

### F.8.4.2 Picture Display Attributes <PDA>

#### F.8.4.2.1 Full Screen Display <FSD>

<FSD> := 2/1 3/0 <ful>

<ful> := 4/5 <length> <value> (Boolean)  
<value> := TRUE;            full screen  
<value> := FALSE;         not full screen

#### F.8.4.2.2 Source ASpect Ratio <ASR>

<ASR> := 2/1 3/1 <araw> <arah>

<araw> := 4/0 <length> <value> (Integer)  
<arah> := 4/0 <length> <value> (Integer)

#### F.8.4.2.3 Photo-area LOCation <LOC>

<LOC> := 2/1 3/2 <loch> <locv>

<loch> := 4/2 <length> <value> (Normalized)  
<locv> := 4/2 <length> <value> (Normalized)

#### F.8.4.2.4 Photo-Area Size <PAS>

<PAS> := 2/1 3/3 <sizw> <sizh>

<sizw> := 4/2 <length> <value> (Normalized)  
<sizh> := 4/2 <length> <value> (Normalized)

#### F.8.4.2.5 Picture PLacement <PPL>

<PPL> := 2/1 3/4 <refh> <refv> <offh> <offv>

<refh> := 4/0 <length> <value> (Integer)  
<refv> := 4/0 <length> <value> (Integer)  
<offh> := 4/2 <length> <value> (Normalized)  
<offv> := 4/2 <length> <value> (Normalized)

#### F.8.4.2.6 Clear Photo-Area <CPA>

<CPA> := 2/1 3/5 <cle>

<cle> := 4/5 <length> <value> (Boolean)  
<value> := TRUE;            clear photo-area  
<value> := FALSE;         do not clear photo-area

### F.8.4.3 Source Picture Attributes <SPA>

#### F.8.4.3.1 Source Picture Comments <PCT>

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

<PCT> := 2/2 3/0 <csz><cid>

<csz> := 4/4 <length> <value> (Enumeration)

<value> :=	0/1	7 bits
	0/2	8 bits
	0/3	16 bits

<cid> := 4/4 <length> <value> (Enumeration)

<value> :=	0/1	non-standard
	0/2	ITU-T Recommendation T.50 [14]
	0/3	T.51 [7]
	0/4	T.61 [8]
	0/5	Data Syntax I
	0/6	Data Syntax II
	0/7	Data syntax III

#### F.8.4.3.2 Source Picture Dimensions <PDS>

<PDS> := 2/2 3/1 <nph> <npv>

<nph> := 4/0 <length> <value> (Integer)

<npv> := 4/0 <length> <value> (Integer)

The 0 pixel value in the <npv> sub-parameter shall be considered as an undefined number of pixel.

#### F.8.4.3.3 Source Pixel Density <PID>

<PID> := 2/2 3/2 <stf> | 2/2 3/2 <p\_h\_num> <p\_h\_den> <p\_v\_num> <p\_v\_den> <unt>

<stf> := 4/4 <length> <value> (Enumeration)

<value> :=	0/1	4:2:2 625 lines
	0/2	4:2:2 525 lines
	0/3	2:1:1 625 lines
	0/4	2:1:1 525 lines
	0/5	CIF
	0/6	Not applicable
	1/0	reserved
	1/1	8 × 3.85 (hor × vert) pixels/mm
	1/2	8 × 7.7 pixels/mm
	1/3	8 × 15.4 pixels/mm
	1/4	16 × 15.4 pixels/mm
	1/5-1/15	reserved
	2/0	200 × 200 pixels/25.4 mm
	2/1	240 × 240 pixels/25.4 mm
	2/2	300 × 300 pixels/25.4 mm
	2/3	400 × 400 pixels/25.4 mm

<p\_h\_num> := 4/0 <length> <value> (Integer)

<p\_h\_den> := 4/0 <length> <value> (Integer)

<p\_v\_num> := 4/0 <length> <value> (Integer)

<p\_v\_den> := 4/0 <length> <value> (Integer)

<unt> := 4/4 <length> <value> (Enumeration)

<value> :=	0/1	pixels/inch
	0/2	pixels/cm
	0/3	pixels/mm

#### F.8.4.3.4 Source SWEEP Direction <SWD>

<SWD> := 2/2 3/3 <sdir> <sdil>

<sdir> := 4/4 <length> <value> (Enumeration)

<value> := 0/1 top to bottom  
0/2 bottom to top

<sdil> := 4/4 <length> <value> (Enumeration)

<value> := 0/1 left to right  
0/2 right to left

#### F.8.4.3.5 DC Images <DCI>

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

<DCI> := 2/2 3/4 <dcv>

<dcv> := 4/5 <length> <value> (Boolean)

<value> := TRUE; DC sized images  
<value> := FALSE; full sized images

#### F.8.4.4 Source Signal Attributes <SSA>

The Source Signal Attributes don't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. They shall not be taken into account if received.

##### F.8.4.4.1 Source Component Description <SCD>

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

<SCD> := 2/3 3/0 <com>

<com> := 4/4 <length> <value> (Enumeration)

<value> := 0/1 R, G, B  
0/2 Y, C<sub>B</sub>, C<sub>R</sub>  
0/3 C, M, Y, K  
0/4 Y

##### F.8.4.4.2 Source Component Data Precision <CDP>

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

<CDP> := 2/3 3/1 <cpt1> <cpt2> <cpt3>.....<cptn>

<cpt> := 4/0 <length> <value> (Integer)

##### F.8.4.4.3 Source Component Order <CMO>

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

<CMO> := 2/3 3/2 <cor1> <cor2> <cor3> ... <corn>

<cor> := 4/0 <length> <value> (Integer)

##### F.8.4.4.4 Source Level Assignment <LAS>

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

<LAS>:= 2/3 3/3 <fix> | 2/3 3/3 <low> <hi>

<fix> := 4/4 <length> <value> (Enumeration)

<value> := 0/1 CCIR Recommendation 601 part 1 [9] levels

<low> := 4/0 <length> <value> (Integer)

<hi> := 4/0 <length> <value> (Integer)

### F.8.4.5 Source Coding Algorithm Attributes <SCA>

#### F.8.4.5.1 JPEG Coding Mode <JPG>

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

<JPG> := 2/4 3/0 <cmp>

<cmp> := 4/0 <length> <value> (Integer)

<value> := val1 + val2 + val3 + val4

val1 := 0 non-hierarchical

1 hierarchical

val2 := 0 DCT

2 DPCM

val3 := 0 sequential

4 progressive

val4 := 0 Huffman coding

8 Arithmetic coding

#### F.8.4.5.2 Encoding Table Management <ETM>

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

<ETM> := 2/4 3/1 <ttp> <tid> <tst> <sfr\_num> <sfr\_den>

<ttp> := 4/4 <length> <value> (Enumeration)

<value> := 0/1 Quantization

0/2 Huffman

<tid> := 4/0 <length> <value> (Integer)

<value> 7/F ALL

NOTE 1 – ALL, depending on the value of <ttp>, indicates all the Quantization tables or all the Huffman tables.

<tst> := 4/4 <length> <value> (Enumeration)

<value> := 0/1 load default table

0/2 use the current table

0/3 table will be transferred

<sfr\_num> := 4/0 <length> <value> (Integer)

<sfr\_den> := 4/0 <length> <value> (Integer)

NOTE 2 – <sfr\_num> and <sfr\_den> may be used only if all the following conditions are fulfilled:

<ttp> is "Quantization"

<tid> is "ALL"

<tst> is "load default table"

Moreover, when used, both <sfr\_num> and <sfr\_den> shall be present and in the order <sfr\_num> followed by <sfr\_den>.

#### F.8.4.5.3 Application Marker codes Assignment <AMA>

This parameter doesn't apply with ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding. It shall not be taken into account if received.

<AMA> := 2/4 3/2 <mak>

<mak> := 4/4 <length> <value> (Enumeration)

<value> := 0/0 X'FFE0' animated images

0/1 X'FFE1' colour palette definition

0/2 to 0/15 is X'FFE2' to X'FFEF' to be allocated

#### F.8.4.5.4 T4 Coding Mode <T4C>

This parameter doesn't apply with ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] coding. It shall not be taken into account if received.

<T4C> := 2/4 3/3 <alg>

<alg> := 4/4 <length> <value> (Enumeration)

<value> :=	0/0	reserved
	0/1	no compression algorithm
	0/2	monodimensional compression algorithm
	0/3	bidimensional compression algorithm with K = 2
	0/4	bidimensional compression algorithm with K = 4
	0/5	reserved

#### F.8.4.5.5 T6 Coding Mode <T6C>

This parameter doesn't apply with ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] coding. It shall not be taken into account if received.

<T6C> := 2/4 3/4 <alg>

<alg> := 4/4 <length> <value> (Enumeration)

<value> :=	0/0	reserved
	0/1	no compression algorithm
	0/2	reserved
	0/3	reserved
	0/4	reserved
	0/5	bidimensional compression algorithm with K = infinite

#### F.8.4.6 Transmission Channel Attributes <TCA>

##### F.8.4.6.1 Translation Mode Encoding <TME>

<TME> := 2/5 3/0 <mod>

<mod> := 4/4 <length> <value> (Enumeration)

<value> :=	0/0	translation mode 0 (no translation)
	0/1	translation mode 1 (no translation except US)
	0/2	translation mode 2 (3 in 4 encoding)
	0/3	translation mode 3 (shift 8 bits)
	0/4	translation mode 4 (shift 7 bits)
	0/5	translation mode 5 (no translation except specific characters)

## F.9 Photographic data

### F.9.1 Introduction

The photographic data shall be sent to the terminal in one or more Picture Entities (PEs)<sup>25)</sup>. The structure and organization of the photographic data is defined by ITU-T Rec. T.81 | ISO/IEC 10918-1 [13].

The photographic data is encoded using all 8 bits of a byte. If only 7 bits of a byte are available for transmitting this data, a translation mechanism shall be selected.

For ITU-T Recommendation T.4 [17] or CCITT Recommendation T.6 [18] coding, and in conformance with ITU-T Recommendation T.563 [21], the first bit of encoded image shall be placed in LSB of octet. The successive bits are placed in the direction of LSB to MSB of octet.

---

<sup>25)</sup> Clause F.5 should be referred to for a description of the use of ISO/IEC 9281 for sending the Photographic Data in a number of PEs.

For ITU-T Recommendation T.4 [17] coding, RTC and EOL shall be used, associated with the tag bit when the two-dimensional coding scheme is used; T.4 fill bits possibly inserted between a line of data and an EOL shall not disturb the T.4 decoder.

For CCITT Recommendation T.6 [18] coding, EOFB shall be used; pad bits possibly used after EOFB shall not disturb the T.6 decoder.

### F.9.2 Translation modes

A number of translation mechanisms are available for sending the photographic data; they are summarised in Table 4.

**Table F.4/T.101 – Translation modes**

Enumeration value	Translation Mode
0	Mode 0, no translation
1	Mode 1, no translation except US
2	Mode 2 (3 in 4 encoding)
3	Mode 3 (shift 8 bits)
4	Mode 4 (shift 7 bits)
5	Mode 5, no translation except specific characters

Modes 2 and 4 can only be used when 7 bits of a byte are available for sending data.

Full transparency, mode 0, is without any translation; the coded data is sent using all 8 bits of a byte.

### F.10 Defaults

Default values of parameters and tables shall be resident in a Videotex terminal supporting photographic mode. If a Videotex terminal receives a photographic header with some parameter values not present, then the current value shall be used. Whether to use default values is indicated by the value of the reset to default parameter. The current value can be either the default value or the value from the last photographic header. Similarly, for Table Data, defaults may be used, the parameter encoding table management shall indicate when default tables shall be used.

#### F.10.1 Default values for photographic header attributes

In Tables F.5 to F.10 default values are provided for all the sub-parameters used in the photographic header.

##### F.10.1.1 Default parameter status attribute

A single parameter, reset to default is used to indicate the status of all other parameters (see Table F.5).

**Table F.5/T.101 – Default parameter status attribute**

Parameter	Sub-parameter	Default value	Comment
Reset To Default <RTD>	<dev>	FALSE	Do not activate default values

### F.10.1.2 Default picture display attributes

The picture display attributes position the photo-area and the photographic image in the physical DDA. The default size for the photo-area and for the photographic image is the full physical DDA. The default of the picture placement parameter defines the top left hand corner of the photo-area and that of the photographic image as the coincident point (see Table F.6).

**Table F.6/T.101 – Default picture display attributes**

Parameter	Sub-parameter	Default value	Comment
Full Screen Display <FSD>	<ful>	FALSE	Not full screen
Source Aspect Ratio <ASR>	<araw> <arah>	4 3	<i>For T.81/ISO 10918-1:</i> Ratio 4:3
	<araw> <arah>	215 297	<i>For T.4/T.6:</i> Ratio 0.724
Photo-area Location <LOC>	<loch> <locv>	0.0 0.0	
Photo-area Size <PAS>	<sizw> <sizh>	1 0.75	The complete DDA
Picture Placement <PPL>	<refh> <refv> <offh> <offv>	0 0 0 0.75	Top left hand corner of image and photo-area
Clear Photo-area <CPA>	<cle>	FALSE	

### F.10.1.3 Default source picture attributes

The default values of the source picture attributes a CIF image (as given in CCITT Recommendation H.261 [6]) (see Table F.7).

**Table F.7/T.101 – Default source picture attributes**

Parameter	Sub-parameter	Default value	Comment
Source Picture Comments <PCT>	<csz> <cid>	0/1 0/3	7 bit T.51
Source Picture Dimensions <PDS>	<nph> <npv>	320 240	<i>For T.81/ISO 10918-1:</i> Minimum resolution image
	<nph> <npv>	1728 1143	<i>For T.4/T.6:</i> A4 page 297 mm × 3.85 lines/mm
Source Pixel Density <PID>	<stf>	0/5	<i>For ISO 10918:</i> CIF
	<stf>	1/1	<i>For T.4/T.6:</i> 8 × 3.85
Source Sweep Direction <SWD>	<sdir> <sdil>	0/1 0/1	Top to bottom Left to right
DC Images <DCI>	<dcv>	FALSE	Full size image

#### F.10.1.4 Default source signal attributes

The default values of the source signal attributes are for a source image with three components (Y, C<sub>B</sub> and C<sub>R</sub>) encoded according to CCIR Recommendation 601, Part 1 [9] (see Table F.8).

**Table F.8/T.101 – Default source signal attributes**

Parameter	Sub-parameter	Default value	Comment
Source Component Description <SCD>	<com>	0/2	Y, C <sub>B</sub> , C <sub>R</sub>
Source Component Data Precision <CDP>	<cpt1> <cpt2> <cpt3>	8 8 8	Precision Y Precision C <sub>B</sub> Precision C <sub>R</sub>
Source Component Order <CMO>	<cor1> <cor2> <cor3>	1 2 3	Y, C <sub>B</sub> , C <sub>R</sub>
Source Level Assignment <LAS>	<fix>	0/1	CCIR Rec. 601, Part 1 [9]

#### F.10.1.5 Default source coding algorithm attributes

The default values of the source coding algorithm attributes are those values which select the baseline system as defined in ISO/IEC DIS 10918-1 [13] (see Table F.9).

**Table F.9/T.101 – Default source coding algorithm attributes**

Parameter	Sub-parameter	Default value	Comment
JPEG Coding Mode Parameter <JPG>	<cmp>	0	a)
Encoding Table Management <ETM>	<tp>	0/1	Quantization
	<tid>		No default
	<st>	0/1	Load default
	<sf_num>	0/1	No scaling factor
	<sf_den>	0/1	No scaling factor
	<tp>	0/2	Huffman
	<tid>		No default
	<st>	0/1	Load default
Application Marker Codes Assignment	<mak>		Not used
T4 Coding Mode <T4C>	<alg>	0/2	Monodimensional
T6 Coding Mode <T6C>	<alg>	0/5	Bidimensional
a) Values which indicate the baseline system.			

#### F.10.1.6 Default transmission channel attributes

See Table F.10.

**Table F.10/T.101 – Default transmission channel attributes**

Parameter	Sub-parameter	Default value	Comment
Translation Mode Encoding <TME>	<mod>	0/0	Full 8-bit transparency

### F.10.1.7 Default application marker code assignment

By default and on reception of a <RTD> command, there is no application marker.

### F.10.2 Default tables

Default tables do not apply for ITU-T Recommendation T.4 [17] and CCITT Recommendation T.6 [18] coding. The applicable tables are given in ITU-T Recommendation T.4 [17] and CCITT Recommendation T.6 [18].

#### F.10.2.1 Default quantization tables

Default quantization tables are given in Tables F.11 to F.16. These quantization tables have been used with good results on 8-bit/sample  $Y$ ,  $C_B$ ,  $C_R$  images.

##### F.10.2.1.1 Default quantization tables for CIF images

See Tables F.11 and F.12.

**Table F.11/T.101 – Default luminance quantization table**

16	11	11	11	12	13	16	20
11	12	12	13	14	16	20	25
12	12	13	14	16	18	27	32
12	14	14	16	21	25	34	42
14	16	18	20	35	40	50	58
14	18	25	35	45	55	65	77
16	20	30	40	50	60	75	85
20	25	35	45	60	72	85	90

**Table F.12/T.101 – Default chrominance quantization table**

17	17	17	18	30	40	50	60
17	20	27	35	60	70	85	99
17	27	30	50	99	99	99	99
18	35	50	60	99	99	99	99
30	60	99	99	99	99	99	99
40	70	99	99	99	99	99	99
50	85	99	99	99	99	99	99
60	99	99	99	99	99	99	99

**F.10.2.1.2 Default quantization tables for 2:1:1 images**

See Tables F.13 and F.14.

**Table F.13/T.101 – Default luminance quantization table**

16	11	11	11	15	17	22	27
12	12	12	12	19	21	29	33
14	14	14	14	22	27	35	40
14	15	16	17	29	35	47	55
19	19	20	22	46	52	62	69
24	27	32	35	64	69	73	84
49	54	60	64	85	90	100	105
72	80	86	92	100	102	104	105

**Table F.14/T.101 – Default chrominance quantization table**

17	17	17	19	30	37	50	62
18	19	20	21	40	58	90	99
24	24	25	26	75	92	99	99
47	50	57	66	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

**F.10.2.1.3 Default quantization tables for 4:2:2 images**

See Tables F.15 and F.16.

**Table F.15/T.101 – Default luminance quantization table**

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

**Table F.16/T.101 – Default chrominance quantization table**

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

**F.10.2.2 Default Huffman tables**

Tables F.17 to F.20 give default Huffman tables which have been developed from the average statistics of a large set of video images with 8-bit precision. In case of no downloading of Huffman tables, they shall be applied to CCIR 4:2:2 images, CCIR 2:1:1 images and CIF images.

**F.10.2.2.1 Default Huffman table for DC luminance differences**

For sequential mode only (see Table F.17).

**Table F.17/T.101 – Default DC luminance differences**

Category	Codelenght	Codeword
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

**F.10.2.2.2 Default Huffman table for DC chrominance differences**

For sequential mode only (see Table F.18).

**Table F.18/T.101 – Default DC chrominance differences**

Category	Codelength	Codeword
0	2	00
1	2	01
2	2	10
3	3	110
4	4	1110
5	5	11110
6	6	111110
7	7	1111110
8	8	11111110
9	9	111111110
10	10	1111111110
11	11	11111111110

**F.10.2.2.3 Default Huffman table for AC luminance coefficients**

For sequential mode only (see Table F.19).

**Table F.19/T.101 – Default table for AC luminance coefficients**

Run/Size	Code length	Code word
0/0	(EOB) 4	1010
0/1	2	00
0/2	2	01
0/3	3	100
0/4	4	1011
0/5	5	11010
0/6	7	1111000
0/7	8	11111000
0/8	10	1111110110
0/9	16	1111111110000010
0/A	16	1111111110000011
1/1	4	1100
1/2	5	11011
1/3	7	1111001
1/4	9	111110110
1/5	11	11111110110
1/6	16	1111111110000100
1/7	16	1111111110000101
1/8	16	1111111110000110
1/9	16	1111111110000111
1/A	16	1111111110001000
2/1	5	11100
2/2	8	11111001
2/3	10	1111110111

**Table F.19/T.101 – Default table for AC luminance coefficients (cont.)**

Run/Size	Code length	Code word
2/4	12	11111110100
2/5	16	111111110001001
2/6	16	111111110001010
2/7	16	111111110001011
2/8	16	111111110001100
2/9	16	111111110001101
2/A	16	111111110001110
3/1	6	111010
3/2	9	111110111
3/3	12	11111110101
3/4	16	111111110001111
3/5	16	111111110010000
3/6	16	111111110010001
3/7	16	111111110010010
3/8	16	111111110010011
3/9	16	111111110010100
3/A	16	111111110010101
4/1	6	111011
4/2	10	1111111000
4/3	16	111111110010110
4/4	16	111111110010111
4/5	16	111111110011000
4/6	16	111111110011001
4/7	16	111111110011010
4/8	16	111111110011011
4/9	16	111111110011100
4/A	16	111111110011101
5/1	7	1111010
5/2	11	11111110111
5/3	16	111111110011110
5/4	16	111111110011111
5/5	16	111111110100000
5/6	16	111111110100001
5/7	16	111111110100010
5/8	16	111111110100011
5/9	16	111111110100100
5/A	16	111111110100101
6/1	7	1111011
6/2	12	111111110110
6/3	16	111111110100110
6/4	16	111111110100111
6/5	16	111111110101000
6/6	16	111111110101001
6/7	16	111111110101010
6/8	16	111111110101011
6/9	16	111111110101100
6/A	16	111111110101101
7/1	8	11111010
7/2	12	111111110111

**Table F.19/T.101 – Default table for AC luminance coefficients (cont.)**

Run/Size	Code length	Code word
7/3	16	111111110101110
7/4	16	111111110101111
7/5	16	111111110110000
7/6	16	111111110110001
7/7	16	111111110110010
7/8	16	111111110110011
7/9	16	111111110110100
7/A	16	111111110110101
8/1	9	11111000
8/2	15	11111111000000
8/3	16	111111110110110
8/4	16	111111110110111
8/5	16	111111110111000
8/6	16	111111110111001
8/7	16	111111110111010
8/8	16	111111110111011
8/9	16	111111110111100
8/A	16	111111110111101
9/1	9	11111001
9/2	16	111111110111110
9/3	16	111111110111111
9/4	16	111111111000000
9/5	16	111111111000001
9/6	16	111111111000010
9/7	16	111111111000011
9/8	16	111111111000100
9/9	16	111111111000101
9/A	16	111111111000110
A/1	9	11111010
A/2	16	111111111000111
A/3	16	111111111001000
A/4	16	111111111001001
A/5	16	111111111001010
A/6	16	111111111001011
A/7	16	111111111001100
A/8	16	111111111001101
A/9	16	111111111001110
A/A	16	111111111001111
B/1	10	111111001
B/2	16	111111111010000
B/3	16	111111111010001
B/4	16	111111111010010
B/5	16	111111111010011
B/6	16	111111111010100
B/7	16	111111111010101
B/8	16	111111111010110
B/9	16	111111111010111
B/A	16	111111111011000
C/1	10	111111010

**Table F.19/T.101 – Default table for AC luminance coefficients (end)**

Run/Size	Code length	Code word
C/2	16	111111111011001
C/3	16	111111111011010
C/4	16	111111111011011
C/5	16	111111111011100
C/6	16	111111111011101
C/7	16	111111111011110
C/8	16	111111111011111
C/9	16	111111111100000
C/A	16	111111111100001
D/1	11	1111111000
D/2	16	111111111100010
D/3	16	111111111100011
D/4	16	111111111100100
D/5	16	111111111100101
D/6	16	111111111100110
D/7	16	111111111100111
D/8	16	111111111101000
D/9	16	111111111101001
D/A	16	111111111101010
E/1	16	111111111101011
E/2	16	111111111101100
E/3	16	111111111101101
E/4	16	111111111101110
E/5	16	111111111101111
E/6	16	111111111110000
E/7	16	111111111110001
E/8	16	111111111110010
E/9	16	111111111110011
E/A	16	111111111110100
F/0	(ZRL) 11	1111111001
F/1	16	111111111110101
F/2	16	111111111110110
F/3	16	111111111110111
F/4	16	111111111111000
F/5	16	111111111111001
F/6	16	111111111111010
F/7	16	111111111111011
F/8	16	111111111111100
F/9	16	111111111111101
F/A	16	111111111111110

**F.10.2.2.4 Default Huffman table for AC chrominance coefficients**

For sequential mode only (see Table F.20).

**Table F.20/T.101 – Default table for AC chrominance coefficients**

Run/Size	Code length	Code word
0/0	(EOB) 2	00
0/1	2	01
0/2	3	100
0/3	4	1010
0/4	5	11000
0/5	5	11001
0/6	6	111000
0/7	7	1111000
0/8	9	111110100
0/9	10	1111110110
0/A	12	111111110100
1/1	4	1011
1/2	6	111001
1/3	8	11110110
1/4	9	111110101
1/5	11	11111110110
1/6	12	111111110101
1/7	16	1111111110001000
1/8	16	1111111110001001
1/9	16	1111111110001010
1/A	16	1111111110001011
2/1	5	1111011
2/2	8	11110111
2/3	10	111110111

**Table F.20/T.101 – Default table for AC chrominance coefficients (cont.)**

Run/Size	Code length	Code word
5/8	16	111111110100100
5/9	16	111111110100101
5/A	16	111111110100110
6/1	7	1111001
6/2	11	1111110111
6/3	16	111111110100111
6/4	16	111111110101000
6/5	16	111111110101001
6/6	16	111111110101010
6/7	16	111111110101011
6/8	16	111111110101100
6/9	16	111111110101101
6/A	16	111111110101110
7/1	7	1111010
7/2	11	1111111000
7/3	16	111111110101111
7/4	16	111111110110000
7/5	16	111111110110001
7/6	16	111111110110010
7/7	16	111111110110011
7/8	16	111111110110100
7/A	16	111111110110101
8/1	8	1111001

**Table F.20/T.101 – Default table for AC chrominance coefficients (end)**

Run/Size	Code length	Code word
B/5	16	111111111010101
B/6	16	111111111010110
B/7	16	111111111010111
B/8	16	111111111011000
B/9	16	111111111011001
B/A	16	111111111011010
C/1	9	11111010
C/2	16	111111111011011
C/3	16	111111111011100
C/4	16	111111111011101
C/5	16	111111111011110
C/6	16	111111111011111
C/7	16	111111111000000
C/8	16	111111111000001
C/9	16	111111111000010
C/A	16	111111111000011
D/1	11	1111111001
D/2	16	11111111100100
D/3	16	11111111100101
D/4	16	11111111100110
D/5	16	11111111100111
D/6	16	11111111101000
D/7	16	11111111101001
D/8	16	11111111101010
D/9	16	11111111101011
D/A	16	11111111101100
E/1	14	1111111100000
E/2	16	11111111101101
E/3	16	11111111101110
E/4	16	11111111101111
E/5	16	11111111110000
E/6	16	11111111110001
E/7	16	11111111110010
E/8	16	11111111110011
E/9	16	11111111110100
E/A	16	11111111110101
F/0	(ZRL) 10	11111010
F/1	15	11111111000011
F/2	16	11111111110110
F/3	16	11111111110111
F/4	16	11111111111000
F/5	16	11111111111001
F/6	16	11111111111010
F/7	16	11111111111011
F/8	16	11111111111100
F/9	16	11111111111101
F/A	16	11111111111110

## F.11 Photographic profiles

This annex contains a data syntax for encoding many different facilities which may be provided by Videotex services supporting photographic mode. In order to ease interworking between photovideotex services and to harmonize these services, a number of service profiles are specified. A service profile determines which specific facilities should be implemented.

Each profile defined below is identified in ITU-T Recommendation T.101 [2] by a specific “TFI” code used for photographic profiles.

Two kinds of profile are defined:

- a) Two groups of compatible profiles are defined, one for sequential image build-up and one for progressive image build-up; this is illustrated in Figure F.11.

Backward compatibility from P5 to P4 to P3 to P2 to P1 is given, i.e. an image elaborated from an information provider according to P1 can be displayed by a terminal which supports any profile P1 to P5. Similarly, an image elaborated according to P4 can only be displayed by a terminal which supports either P5 or P4. P5 and P4 also support progressive build-up of CIF and CCIR 2:1:1 image types.

In ITU-T Recommendation T.101 [2] for the compatible photographic profiles, “TFI” codes are defined for the profiles P1 to P5.

- b) In addition to the profiles described above, in ITU-T Recommendation T.101 [2] a “TFI” code is defined for a “private choice of a photographic profile  $P_{priv}$ ”. The use of that profile allows the selection of any attributes and parameters of this annex. This profile is intended for “specialized private applications” (e.g. use of other colour models, resolutions, precisions), which are out of the scope of the profiles P1 to P5. However, users of this profiles have to assure within their own “private applications” that they can process and display their “private application” images.

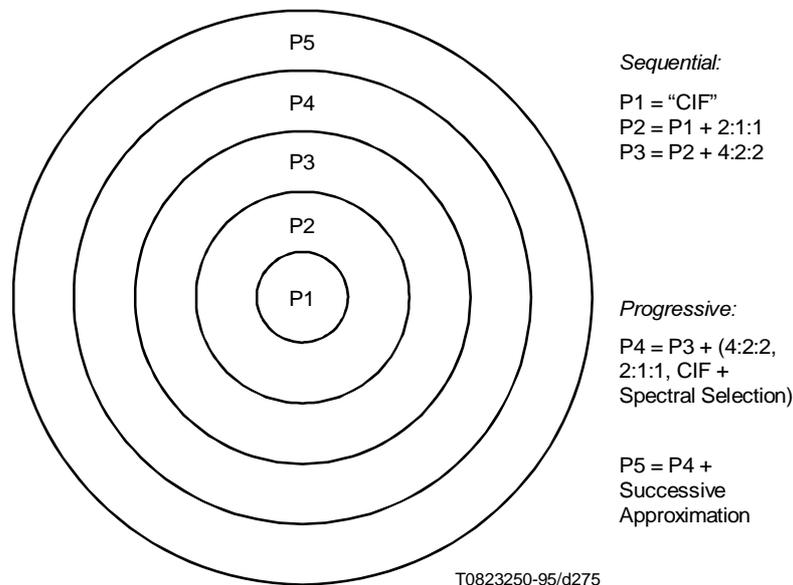


Figure F.11/T.101 – Compatible photovideotex profile structure



**Table F.21/T.101 – Summary of photographic profiles**

Profile Characteristics	P1	P2	P3	P4	P5
Source Pixel Density	CIF	CIF + CCIR 2:1:1	CIF + CCIR 2:1:1 + CCIR 4:2:2	CIF + CCIR 2:1:1 + CCIR 4:2:2	CIF + CCIR 2:1:1 + CCIR 4:2:2
JPEG Coding	Non-hierarchical DCT Sequential Huffman	Non-hierarchical DCT Sequential Huffman	Non-hierarchical DCT Sequential Huffman	Non-hierarchical DCT Sequential and progressive Huffman	Non-hierarchical DCT Sequential and progressive Huffman
Conformance to ITU-T Rec. T.81   ISO/CEI 10918-1	Baseline	Baseline	Baseline	Baseline + extended system, spectral selection	Baseline + extended system, spectral selection and successive approximation

For the definitions of the compatible photographic profiles P1 to P5 and the private choice of photographic profile (P<sub>priv</sub>) the following definitions apply:

**recognize:** Means to determine the syntactic form, but not necessarily the semantics of a code sequence.

**execute:** Means to process a code sequence to allow the display of information conveyed by the code sequence and by subsequent code sequences.

NOTE – The central part of a photovideotex system is the database to which editing equipment and user terminals are connected. Since, for compatibility reasons, some constraints (i.e. the JPEG demand for handling the baseline system) are imposed on the user's terminal which are not relevant for the database, the requirements apply only to user terminals. Requirements on editing equipment are outside the scope of this profile specification.

The issue of interoperability is of major importance. For the photographic data syntax to be truly a common enhancement to all of the base Videotex data syntaxes, it is necessary for images created in one country to be able to be presented on terminals in another country. For example, it should be possible to display a common photographic image on a DS III Videotex terminal in North America over a DS III background image, on a DS I Videotex terminal in Japan over a DS I background image or on a DS II Videotex terminal in Europe over a DS II background image. Fortunately the burden of the different Videotex systems can be quite small. For instance, an image defined in North America using the minimum DS III SRM resolution of 256 × 200 could be displayed on a DS II terminal in Europe by displaying the image in a 256 × 200 subset of the 320 × 240 base European display screen resolution (shrinking technique). A similar approach can be taken for the 248 × 204 base Japanese DS I resolution. Conversion onto DS I or DS III terminals would probably make use of scaling the image since such a scaling is inherent in the geometric principles used in DS III and I (higher ranks) (see Table F.22).

Techniques for performing such rendering are described in Appendix II of this annex.

**Table F.22/T.101 – Size of physical DDA**

Profile	P1	P2	P3	P4	P5
DS I	248 × 204	496 × 408	Further study	Further study	Further study
DS II	320 × 240	320 × 480	640 × 480	640 × 480	640 × 480
DS III	256 × 200	256 × 400	640 × 480	640 × 480	640 × 480

### F.11.1 Compatible photographic profiles (P1 to P5)

In the following subclauses the five compatible photographic profiles are described. The profile is defined by stating which parts of this annex shall be supported and which parts of ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] shall be supported.

#### F.11.1.1 Profile P1

This is the profile for a service with the pixel density defined in CCITT Recommendation H.261 [6] for CIF. It only uses the baseline system – for sequential mode of display – as defined in ITU-T Rec. T.81 | ISO/IEC 10918-1 (“JPEG”) [13].

The profile shall satisfy the following clauses and subclauses of this annex:

F.5 ISO/IEC 9281 syntax and switching structure: the described syntax and switching structure shall be recognized and executed.

F.6 Coding of the picture data entity: the described coding shall be recognized and executed.

F.7.3.1.1 Reset to default: shall be recognized and executed.

F.7.3.2.1 Full Screen display: shall be recognized and executed.

NOTE 1 – This is a local matter in the terminal. Correct rendering of the full screen cannot be guaranteed.

F.7.3.2.2 Source aspect ratio: shall be recognized, but only the value 4/3 shall be valid.

F.7.3.2.3 Photo-area LOCation: shall be recognized and executed (inside the physical DDA).

F.7.3.2.4 Photo-area size: shall be recognized and executed (less than or equal to the physical DDA).

F.7.3.2.5 Picture PLacement: shall be recognized and executed with the Reference Point in the top left-hand corner of the picture (inside the physical DDA).

NOTE 2 – For some implementations, when used in conjunction with a one-layer alphamosaic display, location and size of the photographic image may be restricted by the mosaic character positions, e.g. multiple of 8 pixels horizontally and multiple of 10 pixels vertically.

F.7.3.2.6 Clear photo-area: shall be recognized and executed.

F.7.3.3.1 Source picture comments: shall be recognized, but use is optional.

F.7.3.3.2 Source picture dimensions: shall be recognized and executed.

F.7.3.3.3 Source PIXel density: shall be recognized, but only CIF shall be valid.

F.7.3.3.4 Source SWEEP direction: shall be recognized, but only line-build-up: left to right and row-build-up: top to bottom shall be executed.

F.7.3.3.5 DC images: shall be recognized and executed.

F.7.3.4.1 Source component description: shall be recognized, but only Y or Y, C<sub>B</sub>, C<sub>R</sub> with block interleave shall be executed.

F.7.3.4.2 Source component data precision: shall be recognized, but only up to 8 bits per component shall be executed.

F.7.3.4.3 Source component order: shall be recognized, but only the default shall be executed.

F.7.3.4.4 Source level assignment: shall be recognized, but only CCIR Recommendation 601, Part 1 [9] shall be valid.

F.7.3.5.1 JPEG coding mode: shall be recognized:

- for hierarchical mode only non-hierarchical mode shall be executed;
- for type of algorithm only DCT shall be executed;
- for build-up mode only sequential mode shall be executed;
- for entropy coding technique only Huffman coding shall be executed.

F.7.3.5.2 Encoding table management: shall be recognized, and shall be executed except <ttp> = “conditioning tables”. The number of loaded tables is limited to one set, consisting of Quantization tables and Huffman tables, for profiles 1, 2 and 3.

F.7.3.5.3 Application marker codes assignment: shall be recognized. Use of the colour palette marker code is optional; when it is used, the formatting shall conform to the clause on colour definition of this Recommendation.

F.7.3.6.1 Translation mode encoding: shall be recognized.

The following translation mechanisms shall be used:

- mode 0 (mandatory);
- mode 4 (optional);
- mode 2 (see Note 3).

NOTE 3 – The mode 2 is required in all cases, except for close systems (including external data bases) which guarantee 8-bit data transparency.

The full conformance to the baseline system described in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] shall be valid.

#### **F.11.1.2 Profile P2**

This is the profile of a service with the pixel density defined by the CCIR Recommendation 601, Part 1 [9], in the level “2:1:1” of the compatible family (see Note below). It only uses the baseline system – for sequential mode of display – as defined in ITU-T Rec. T.81 | ISO/IEC 10918-1 (“JPEG”) [13].

NOTE – The CCIR 2:1:1 is a straight forward derivation of the CCIR 601 4:2:2. It is not fully described in CCIR Recommendation 601, Part 1 [9], so refer to Appendix V for a full description according to “CCIR Recommendation 601 [9]” rules.

The profile shall satisfy the following clauses and subclauses of this annex:

Profile P1 plus the following:

F.7.3.3.3 Source Pixel density: shall be recognized, but only CIF and CCIR 601 [9] 2:1:1 shall be valid.

The full conformance to the baseline system described in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] shall be valid.

#### **F.11.1.3 Profile P3**

This is the profile of a service which implements normal resolution TV pictures (CCIR Recommendation 601, Part 1, 4:2:2 [9]) with sequential picture build-up.

The profile shall satisfy the following clauses and subclauses of this annex:

Profile P2 plus the following:

F.7.3.2.5 Picture Placement: shall be recognized, the default shall be executed, execution of other values shall be optional.

F.7.3.3.3 Source Pixel density: shall be recognized, but only CIF, CCIR 601 2:1:1 and CCIR 601 4:2:2 shall be valid.

The full conformance to the baseline system described in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] shall be valid.

#### **F.11.1.4 Profile P4**

This is the profile of a service which implements normal resolution TV pictures (CCIR Recommendation 601, Part 1, 4:2:2) [9] with both sequential and progressive (spectral selection only) picture build-up.

The profile shall satisfy the following clauses and subclauses of this annex:

Profile P3 plus the following:

F.7.3.3.3 Source Pixel density: shall be recognized, but only CIF, CCIR 601 2:1:1 and CCIR 601 4:2:2 shall be valid.

F.7.3.5.1 JPEG coding mode: shall be recognized:

- for hierarchical mode only non-hierarchical mode shall be executed;
- for type of algorithm only DCT shall be executed;
- for build-up mode both sequential and progressive mode shall be executed;
- for entropy coding technique only Huffman coding shall be executed.

The full conformance to the baseline system described in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] shall be valid.

Conformance to the extended system for progressive picture build-up using spectral selection only as described in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] shall be valid.

NOTE – For this profile, the technique described in Appendix III (“special spectral selection”) can be used to allow compatibility between different resolution terminals.

#### **F.11.1.5 Profile P5**

This is the profile of a service which implements normal resolution TV pictures (CCIR Recommendation 601, Part 1, 4:2:2 [9]) with both sequential and progressive (both spectral selection and successive approximation) picture build-up.

The profile shall satisfy the following clauses and subclauses of this annex:

Profile P4 plus the following:

The full conformance to the baseline system described in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] shall be valid.

Conformance to the extended system for progressive picture build-up using spectral selection and successive approximation as described in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] shall be valid.

#### **F.11.2 Private choice of photographic profile ( $P_{priv}$ )**

This profile gives users of “private applications” the flexibility to choose any attributes and parameters from this annex (i.e. it allows the definition of applications that are out of the scope of the compatible photographic profiles described in subclause F.11.1).

NOTE – Some examples of “private use applications” could be:

- an HDTV image coded with DCT and Arithmetic Coding for professional use;
- a 2000 × 2000 C, M, Y, K image with DCT and Huffman coding for a specific printing application;
- a 16 bits/pixel 512 × 512 luminance only image with lossless DPCM for certain medical applications.

Since the same “TFI” code shall be used for all types of “private application”, there shall be no guarantee provided for compatibility among all users of “private use applications”. For this reason, to achieve compatibility in “private use applications”, users of a given application have to reach mutual agreement within their “private application” on the specific attributes and parameters used.

Terminals built according to the profile  $P_{priv}$  shall recognize all the attributes and parameters defined by this annex, but whether they are actually executed is not defined by this profile.

The full conformance to the baseline system, as described by ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] shall be valid. Conformance to the extended system, as described by ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] might be applicable to some “private applications”. The relevance of this is outside the scope of this annex.

### **F.12 Combination rule**

Combination of photographic data encoded following this annex and other types of information encoded following any of the data syntaxes (DSI, DSII and DSIII) is dependent on the image built up rule defined in the various data syntaxes (DSI, DSII and DSIII). If no rule is specified, time dependent (last arrived, last displayed) sequential built up is assumed.

**Appendix I**  
(Informative)  
(to Recommendation T.101, Annexe F)

**Photovideotex tutorial**

**I.1 Introduction**

For several years a standardization effort known by the name JPEG (Joint Photographic Experts Group) has been working toward establishing the first international digital image compression standard for continuous-tone (multi-level) still images, both greyscale and colour, for very broad applications comprising computer applications and telecommunication services.

The “Joint Photographic Expert Group”, JPEG, was established in 1986 as a joint committee, under the auspices of both ISO IEC/JTC1/SC2/WG8 (coded representation of picture and audio information) and CCITT/SGVIII (Q.16), for the purpose of standardizing colour image compression techniques. In 1990, the ad-hoc group “JPEG” became a new working group (ISO IEC/JTC 1/SC2/WG10), with WG8 taking on a planning and coordination role.

ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] describes this algorithm.

Videotex service providers will certainly be able to take advantage of this technique by implementing an applicative syntax for the photographic mode of Videotex.

This appendix describes the basic principles used by the JPEG algorithm in the compression of a photographic image. The aim is to give the reader a basic understanding of greyscale and continuous colour picture compression, and thus does not include all the technical details necessary to implement the algorithm.

**I.2 The present state of photovideotex**

All three of the existing base Videotex data syntaxes support some form of a photographic mode.

Data Syntax I is basically an alpha-photographic (character and pattern) approach; however, the definition of attributes is not specified on pixel boundaries and there is no effective compression, therefore the new JPEG based algorithms are considered complementary to the basic photographic mode defined in Data Syntax I.

Part 3 of Annex C to ITU-T Recommendation T.101 [1] contains the photographic syntax for services existing at the beginning of 1990. There are currently two coding techniques in use:

- 1) Differential Pulse Code Modulation (DPCM) image coding;
- 2) Discrete Cosine Transformation (DCT) image coding (CEPT 1984).

Although these two techniques are adequate to encode photographic data, they do not provide enough compression (ratio of input data to output data for the compression process) for large scale Photovideotex development. The new algorithm developed by JPEG, based on the DCT, provides a considerable improvement compared with the currently used techniques (note that JPEG also describes an algorithm for lossless compression, based on a DPCM, see later). As an example, a natural CCIR 601 4:2:2<sup>26</sup>) colour image is compressed by a factor in the order of 20 with the JPEG algorithm compared to the currently achieved ratio of 10. Such an image needs approximately 830 Kbytes for uncompressed storage. By applying the DCT compression techniques developed by JPEG this can be reduced typically to 40 Kbytes.

The JPEG algorithm offers sufficient flexibility to accommodate a wide variety of pixel resolutions, colour spaces and transmission bandwidths.

The JPEG algorithm also offers to display reconstructed images in two ways, known as sequential mode and progressive mode. In sequential mode the image is coded or decoded to full quality in one pass, from the top to the bottom of the screen. In progressive mode the entire image is treated in separate passes, the displayed image becoming progressively sharper.

---

<sup>26)</sup> CCIR 601 resolution for the luminance component is  $720 \times 576$ .

The JPEG compression algorithm described in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] consists of three basic functional parts:

- a “baseline sequential system”;
- an “extended system” for greater precision and progressive coding modes; and
- an “independent function” for sequential lossless (reversible) coding.

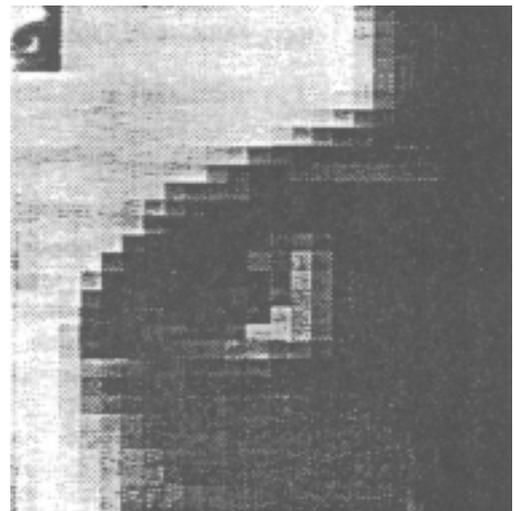
The baseline system will form the basis for photovideotex applications, it is the minimum implementation required by all DCT-based systems defined in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13]. The baseline system is described in more detail in I.5.

### I.3 Image representation

The ways one can obtain an image in a discrete form, are varied and constitute usually a specific domain of digital signal processing. However, this is not the subject of this tutorial; the reader is referred to the few selected books given in the bibliography subclause of this appendix (I.8).

In general terms, the photographic image is usually defined as a set of 2-dimensional tables where each table represents an array of numbers (often natural integers), each number representing in turn a value of a sample of a given component. As illustrated in Figure I.2, a source image consists of a set of  $N_c$  components (e.g.  $N_c = 3$ , for the “R, G, B” colour space), with overall dimensions  $X$  pixels horizontally (pixel = picture element, or the smallest element of the picture) by  $Y$  pixels vertically. Each component is represented by a rectangular array of samples of dimension  $x_i$  by  $y_i$ . The number of samples of each component is not necessarily identical. We are referring here to a sub-sampling processing (e.g. in the  $Y, C_B, C_R$  colour space, the  $C_B$  and  $C_R$  components are sub-sampled by a factor of 2 compared to the  $Y$  component).

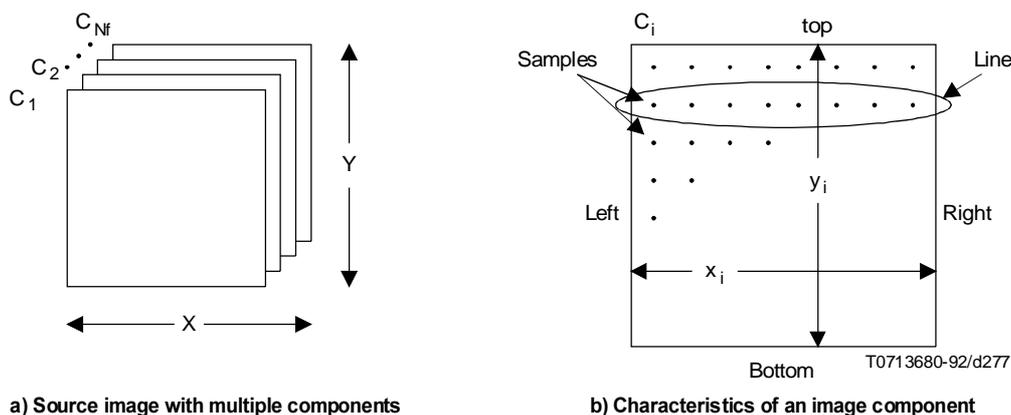
From this representation, when reconstructing the image for display, each pixel shall be shown as a small rectangle filled with a colour calculated from the samples values of each component which represent the pixel. Figure I.1 gives a practical illustration of the above definitions.



T0823260-95/d276

Pixel array of a portion of the image

Figure I.1/T.101 – Illustration of a digitized image



**Figure I.2/T.101 – Image formal representation**

As examples, typical array dimensions are (for luminance):

- 320 × 576 (CCIR 2:1:1);
- 640 × 350 (EGA with border);
- 640 × 480 (VGA with border);
- 720 × 576 (CCIR 4:2:2).

The image in digital form is then suitable for further digital processing. The algorithms developed by JPEG are applied to the digital image to compress the amount of stored data and to make the transport and storage of the image data more efficient.

## **I.4 The JPEG compression technique**

### **I.4.1 Lossy and lossless compression**

The JPEG standard specifies two classes of compression processes, lossy and lossless processes. Those based on the Discrete Cosine Transform (DCT) are lossy, thereby allowing substantial compression to be achieved while producing a reconstructed image with high visual fidelity to the encoder's source image.

The simplest DCT-based compression process is referred to as the baseline sequential process. It provides a capability which is sufficient for many applications. There are additional DCT-based processes which extend the baseline sequential process to a broader range of applications. In many application environments which use extended DCT-based decoding processes, the baseline decoding process is required to be present in order to provide a default decoding capability.

The second class of compression process is not based upon the DCT and is provided to meet the needs of applications requiring lossless compression. This lossless compression scheme is used independently of any of the DCT-based processes.

The amount of compression provided by the various processes is dependent on the characteristics of the particular image being compressed, as well as on the picture quality desired by the application.

For colour images with moderately complex scenes, all DCT-based modes of operation typically produce the following levels of picture quality for the indicated ranges of compression. These levels are only a guide-line – quality and compression can vary largely according to source image characteristics and scene content:

- 0.25-0.5 bits/pixel: moderate to good quality, sufficient for some applications;
- 0.5-0.75 bits/pixel: good to very good quality, sufficient for many applications;

- 0.75-1.5 bits/pixel: excellent quality, sufficient for most applications;
- 1.5-2.0 bits/pixel: usually indistinguishable from the originals, sufficient for the most demanding applications.

The lossless process produces from 1.5:1 to 4:1 compression ratios, this means about 4 bits/pixel for a CCIR 601 4:2:2 colour image.

#### I.4.2 Modes of encoding

There are four main modes of encoding available:

- Sequential* – Each component of the image is encoded in a single left-to-right, top-to-bottom scan.
- Progressive* – The image is encoded in multiple scans, giving first a crude image refined by additional scans.
- Hierarchical* – The image is encoded with multiple spatial resolutions, for adaptation to network speed and to terminal resolution capability. Typically, the lower-resolution versions of the image may be accessed without having to decompress the source image's full resolution.

In hierarchical mode an image is encoded as a sequence of frames. These frames provide reference reconstructed components which are usually needed for prediction in subsequent frames. Except for the first frame of a given component, differential frames encode the difference between source components and reference reconstructed components. The coding of the differences may be done using only DCT-based processes, only lossless processes, or DCT-based processes with a final lossless process for each component. Downsampling and upsampling filters may be used to provide a pyramid of spatial resolutions as shown in Figure I.4. Alternatively, the hierarchical mode can be used to improve the quality of the reconstructed components at a given spatial resolution.

There exist different ways for implementing the hierarchical mode, one of them is described in Appendix III.

- Lossless* – An encoding shall will guarantee exact recovery of the source image, for an acceptable compression ratio.

All those modes can be combined with however some limitations (see ITU-T Rec. T.81 | ISO/IEC 10918-1 [13]).

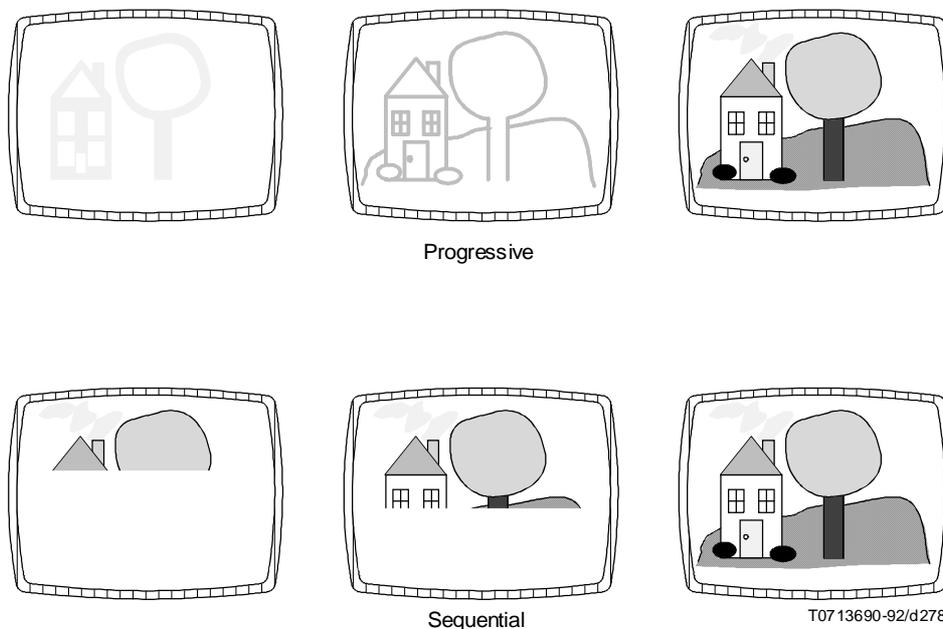
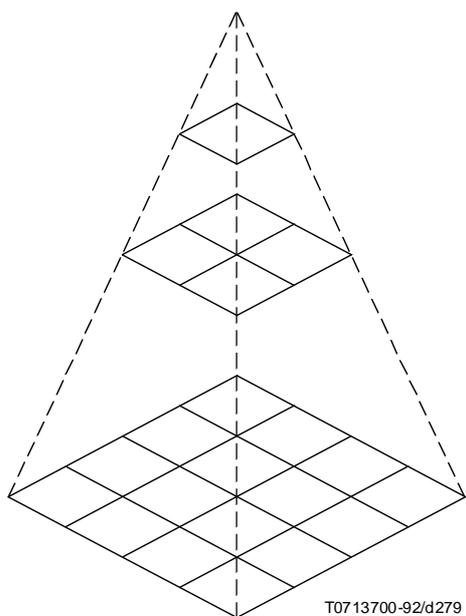


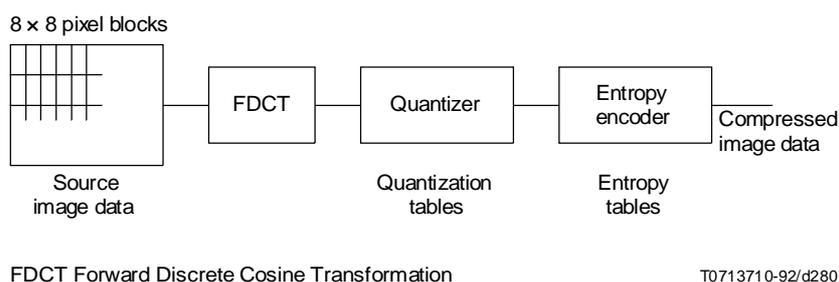
Figure I.3/T.101 – Progressive versus sequential presentation



**Figure I.4/T.101 – Hierarchical multi-resolution encoding**

### I.4.3 The DCT-based coding

The JPEG compression algorithm described in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] makes use of a variety of techniques to achieve compression. Figure I.5 illustrates the key processing steps performed by all modes of encoding of the DCT-based algorithm (the only mode of encoding considered here) in the form of a block diagram.



**Figure I.5/T.101 – DCT-based encoder functional blocks**

Common to all encoding modes of the DCT-based algorithm an  $8 \times 8$  DCT is applied to the image data to concentrate the information present in the blocks of  $8 \times 8$  component samples into a relatively small number of transform coefficients. The data is then quantized to reduce the number of bits used to encode the coefficients and Huffman coding or arithmetic coding is used to achieve further compression before the data is transmitted or stored.

A description of these techniques and the way in which they are applied by the JPEG algorithm is given in the following subclauses. In this appendix only, Huffman coding will be described (for the description of the arithmetic coder see the ITU-T Rec. T.81 | ISO/IEC 10918-1 [13]).

#### I.4.3.1 The discrete cosine transform

The discrete cosine transform is used for its good property of *energy compaction* when applied to an image signal. When the forward DCT (FDCT) is applied to a block of picture samples it is transformed to a block of coefficients. Each coefficient representing the “weight” of a given  $8 \times 8$  pattern, so-called “*cosine image-basis*” (see Figure I.6), present in the given  $8 \times 8$  image samples. It can also be viewed as “the importance” of a *spectral band* in the original block of  $8 \times 8$  image samples. The coefficient with zero frequency in both dimensions is known as the DC coefficient, and the other 63 are known as the AC coefficients. It is important to note that this transformation provides *no compression* at all, simply it changes the mode of representation of the source image, as does any linear transformation. However, a property of the FDCT is that, typically and statistically, it concentrates the energy contained in the block of picture samples into just a few of the transform coefficients. It is this compaction of energy property that finally allows later at the compression stages (quantization and entropy coding) good compression.

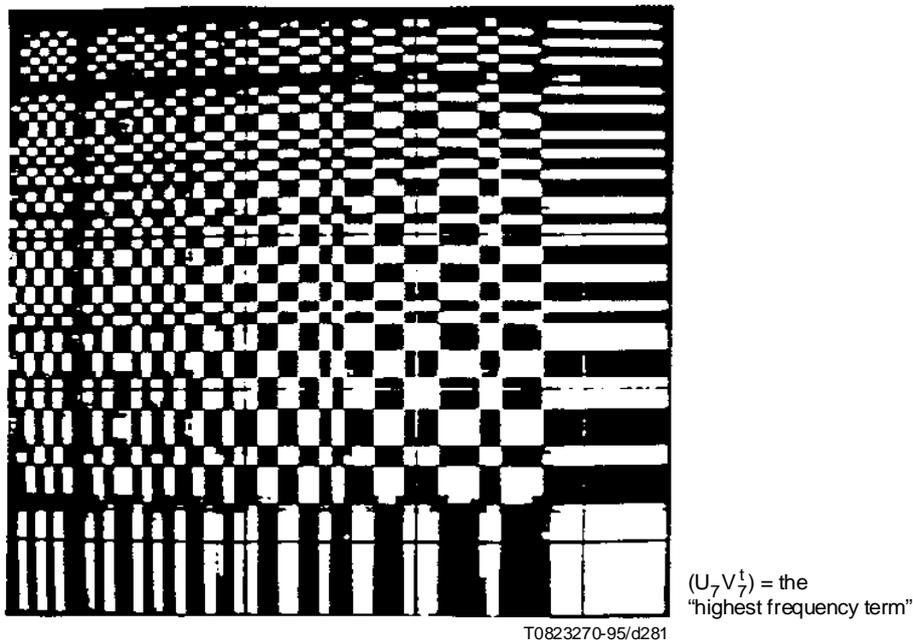


Figure I.6/T.101 – The  $8 \times 8$  DCT basis sub-images

For the DCT-based algorithm the image to be transformed is divided into  $8 \times 8$  pixel blocks. The FDCT is applied to an  $8 \times 8$  block of picture samples, this results in an  $8 \times 8$  block of coefficients. The first coefficient termed the DC value is eight times the average of the 64 pixel values. Each of the remaining coefficients, referred to as  $AC_1$  to  $AC_{63}$ , correspond to a two dimensional “DCT-filtered” signal within the pixel block. Due to the previously mentioned statistical properties, typically only a few of the 63 AC coefficients have a non-zero value.

A formalization of the process is now outlined.

A general separable linear transformation on an image matrix  $[f]$  may be written in the form:

$$[F] = [U]^t [f] [V] \quad (1)$$

where  $[F]$  is the transform of the image  $f(i,j) = [f]$ ,  $[U]$  and  $[V]$  are the operators, and the subscript  $t$  denotes the matrix transpose.  $[U]$  and  $[V]$  are orthogonal operators if:

$$\begin{aligned} [U]^t [U] &= I \\ [V]^t [V] &= I \end{aligned}$$

Hence, the inverse transform of (1) may be written in the form:

$$[f] = [U] [F] [V]^t$$

If the operators  $[U]$  and  $[V]$  are written in the form of column-vectors:

$$\begin{aligned} [U] &= [\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_n] \\ [V] &= [\mathbf{v}_1 \ \mathbf{v}_2 \ \dots \ \mathbf{v}_n] \end{aligned}$$

then:

$$[F] = [\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_n] [f] [\mathbf{v}_1 \ \mathbf{v}_2 \ \dots \ \mathbf{v}_n]^t$$

it follows that:

$$[f] = F_{ij} \mathbf{u}_i \mathbf{v}_j^t$$

This means that the original image  $[f]$  is a weighted (“coefficient”  $F_{ij}$ ) sum of “DCT-subimages” ( $\mathbf{u}_i \mathbf{v}_j^t$ ). The weight  $F_{ij}$  giving the importance of a given subimage ( $\mathbf{u}_i \mathbf{v}_j^t$ ) in the original image  $[f]$ . The terms ( $F_{ij}$ ) are the so-called DCT coefficients and the subimages are the 2-D representation of the DCT transformation kernel. Those 64 DCT-subimages ( $\mathbf{u}_i \mathbf{v}_j^t$ ) are shown below.

$$(U_0 V_0^t) = \text{“DC term”} \quad (U_0 V_7^t)$$

In order to simplify the implementation, the DC value is centred around zero by subtracting  $2^{(P-1)}$  before the DCT is applied.  $P$  is the precision of the unsigned array elements in bits (8 for the baseline system). After the inverse DCT, the level shift is removed. Thus, both DC and AC can use the same 2’s complement representation.

The cosine transform is intrinsically reversible, i.e. theoretically, after a cosine transform, an inverse transform reproduces the pixel values from the transform coefficients. However, an actual implementation may cause non-reversibility due to calculation noise and differences in the type of algorithm used.

For the mathematical description of the DCT see ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] and informative references [I.4] and [I.5] given in I.8 (Bibliography).

#### I.4.3.2 Quantization

The next step in the encoding process is the quantization of the  $8 \times 8$  block containing the DCT coefficients. This step constitutes the main compression stage, the lossy stage and so is the most “crucial” point of the algorithm.

It is worth stressing that loss is only introduced during the quantization process, i.e. any artifacts which are visible in the decompressed image are due to the quantization stage, which makes the whole compression scheme irreversible.

Depending on the viewing conditions, tests with human observers have shown that AC coefficients are not of equal importance. This “psychovisual” property allows for coarser quantization of the “less sensitive” coefficients and finer quantization of the “more sensitive” ones. Moreover, according to the application (i.e. type of images, viewing conditions, type of application, etc.) the sensitivity law is drastically different! Furthermore, the derivation of the “optimum” quantization steps is not a straightforward task! So, each application has to define its own quantization

values, nearly as “proprietary” values, since no default values are applicable. Typically in this annex for the photovideo service, we recommend using a proposed set of quantization tables which have been tested. However, as the values are systematically sent with the compressed image, any other values could be used.

The quantization values are specified in an  $8 \times 8$  table (quantization table), each of the DCT coefficients has a separately specified quantization value. The linear quantization procedure is as follows:

- $Q(u,v)$  represents the table of quantization values indexed by  $u$  and  $v$ ;
- $F(u,v)$  represents the unquantized DCT coefficients;
- $F'(u,v)$  represents the dequantized DCT coefficients;
- $C(u,v)$  represents the quantized DCT coefficients.

For  $F(u,v) \geq 0$

$$C(u,v) = \{F(u,v) + Q(u,v)/2\} / Q(u,v)$$

and for  $F(u,v) < 0$

$$C(u,v) = \{F(u,v) - Q(u,v)/2\} / Q(u,v)$$

The dequantization process is also linear:

$$F'(u,v) = C(u,v) \times Q(u,v)$$

In principle a different quantization table should be defined for each colour coordinate system, spatial resolution, data precision and application. Examples of tables which produce good results for CCIR 601 4:2:2 images are given in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13], and examples of tables which produce good results for CCIR 601 2:1:1 and ITU-T “CIF” are given in this annex. Two quantization tables are used, one for the luminance components and one for the chrominance components.

For example, the precision of the input data for the baseline system is restricted to 8 bits  $[-128, 127]$  and consequently the quantized transformed coefficients are limited to 11 bits  $[-1024, 1023]$  (i.e. if the quantization step equals 1).

After quantization, the DCT coefficients of the baseline system have a precision as follows:

$$11 - \log_2(M)$$

where  $M$  is the quantization value for the coefficient.

#### **I.4.3.3 Huffman coding**

Huffman coding is a form of entropy coding. Entropy coding techniques achieve compression by exploiting the redundancy that occurs in the data they encode. Redundancy is a characteristic which is related to “predictability” in the data. For example, an image of constant grey level is fully predictable once the grey level of the first pixel is known. On the other hand, a white noise random field is totally unpredictable and every pixel has to be stored to reproduce the image.

The aim of entropy coding is to encode a block of data with a bit rate which is close to the entropy of the block. Consider a block containing  $M$  transformed coefficients with each coefficient represented by  $B$ -bits. The block contains a total of  $MB$  bits with probabilities  $p_i$ . The entropy of the block is given by:

$$p_i (-\log_2 p_i) = H$$

where  $i = 0, 1 \dots 2^{MB}$ .

If the data contains redundancy, the entropy shall be less than  $B$  bits per coefficient.

Entropy coding results in variable length coding, i.e. highly probable coefficient values are represented by small-length codes, and vice versa.

Huffman coding is the technique applied to the AC and DC transform coefficients to assign a short code word to the most frequently occurring configuration of coefficients.

In the JPEG DCT-based algorithm, the quantized DC coefficient is treated separately from the 63 AC coefficients. Differential Pulse Code Modulation (DPCM) is used to encode the DC value prior to Huffman coding. Zigzag ordering is applied to the AC coefficients prior to Huffman coding.

**I.4.3.3.1 PCM encoding of the DC coefficients**

Since there usually exists a strong correlation between DC coefficients of adjacent 8 × 8 blocks, the DC coefficient of a given block is differentially encoded with respect to that of the previous block.

For a given block i, this is expressed by the relationship:

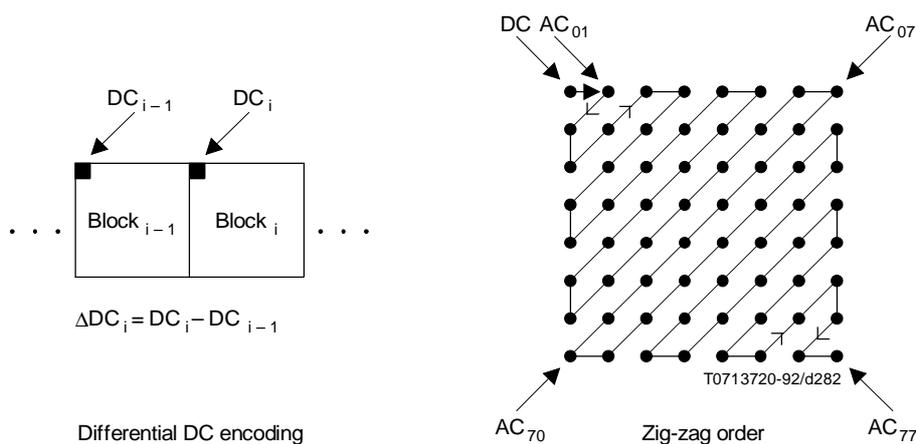
$$\text{Diff\_DC}(i) = \text{DC}(i) - \text{DC}(i - 1)$$

In the decoder, the difference is decoded and added to the prediction. At the start of processing, the value 0 is used for the prediction. As explained in I.4.3.1, the DC values are level shifted and centred around zero.

**I.4.3.3.2 Zigzag ordering of AC coefficients**

For the purpose of entropy coding, the two-dimensional array of DCT coefficients is re-arranged using a zigzag scan into a one-dimensional array. The zigzag ordering of the coefficients is illustrated in Figure I.7.

The coefficients are ordered so that the “low frequency” coefficients occur first. This scanning follows the statistical property of compaction of the energy “around the DC coefficient” where the coefficients are usually non-zero around the DC coefficient and zero for the high frequency AC. So, to increase the compression, AC values and runs of zeros are grouped and coded as (0 runs, AC value). In addition if the remaining ACs of the 8 × 8 blocks are zeros, there are encoded explicitly as an EOB code (End of Block).



**Figure I.7/T.101 – Zigzag ordering of the AC coefficients**

#### I.4.4 Lossless coding

The main procedure for the lossless encoding process is depicted in Figure I.8. A predictor combines the values of up to three neighbouring samples (A, B, and C) to form a prediction of the sample indicated by X in Figure I.8. This prediction is then subtracted from the actual value of sample X, and the difference is losslessly entropy coded by either Huffman or arithmetic coding.

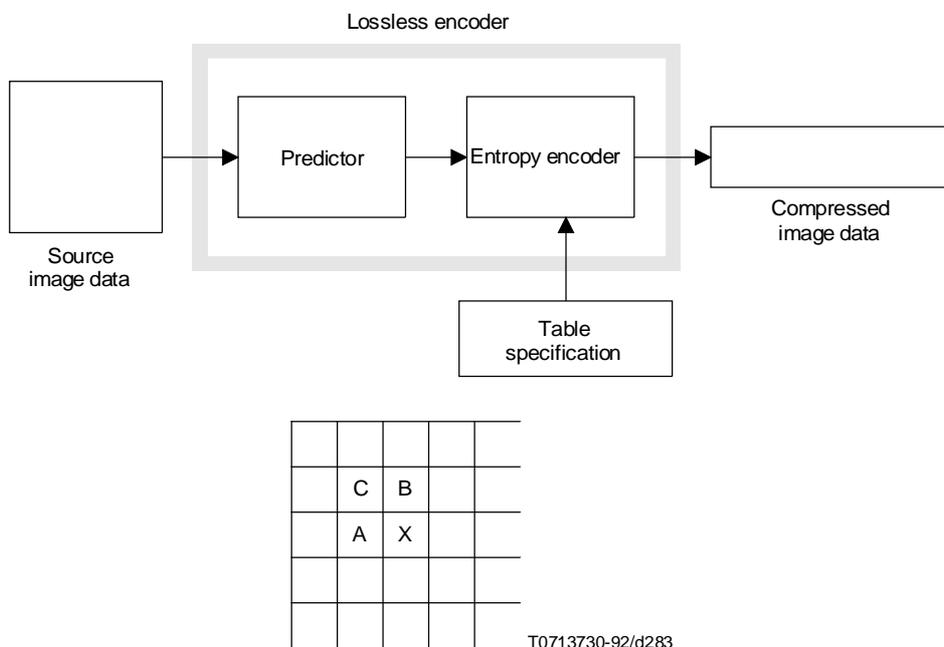


Figure I.8/T.101 – Lossless encoder schematic

#### I.4.5 Source images and data interleaving

The previous subclauses described how a single component of an image is decomposed in  $8 \times 8$  blocks and then transformed, quantized, and coded in a DCT-based mode of operation. Although the JPEG algorithm is able to treat only a single component image (monochrome or greyscale) it is also designed to handle a wide range of multiple-component images.

A multiple component image may be compressed in either interleaved or non-interleaved order. In the latter case each image component is entirely compressed and output as part of the compressed data stream in a separate scan before starting the next component. In the interleaved case,  $8 \times 8$  blocks from each component are processed in a round-robin-fashion, and compressed blocks are output in interleaved order into the data stream. A maximum of four components may be interleaved within a scan.

In the sequential mode both orders of interleave are allowed. As an example a three components image, such as a CCIR 601 4:2:2 image, should be compressed in three separate scans in non-interleaved mode but only in a single scan if the image components are interleaved. A mixture of interleaving order is also permitted, e.g. for the CCIR 601 4:2:2, the luminance Y can be coded in a single non-interleave scan while the chrominance components can be coded together in a single interleave scan.

In the progressive mode only the DC coefficients of different components can be interleaved in the same scan, AC coefficients have to be compressed in a separate scan for each component.

#### **I.4.6 Data organization and signalling parameters**

The encoded data stream contains not only the compressed image data but also parameters which contain information about the data which is being sent. For example, codes to indicate the start of the image or the start of a table are contained in the data stream.

A detailed discussion of these parameters is not included in this annex. The normative part of this annex contains a description of the data organization and signalling parameters as applicable to photovideotex applications. Other issues which are relevant for an implementation of the JPEG algorithm in the photovideotex environment, such as the use of tables (quantization and entropy tables), are also described in the normative part of this annex.

#### **I.5 The baseline system**

The simplest DCT-based coding process is referred to as the baseline sequential process. It is required to be present in all DCT-based modes of operation for compliance to JPEG. It provides a capability which is cost-effective and sufficient for many applications and it acts as a “fall-back” or default mode, for international interworking between services.

The baseline sequential system codes an image to full quality in one pass, the processing usually starting at the top of the image and finishing at the bottom. At each point only a small part of the image is being buffered, the sequential mode is therefore suitable for applications without a full image buffer.

The DCT is applied in the sequential mode as described in I.4.2. The transform coefficients are quantized as explained in I.4.3.2. Following quantization and ordering the coefficients undergo Huffman encoding.

The baseline sequential system has the following main characteristics:

- a) it operates on images with 8 bits/pixel/component only;
- b) it uses Huffman coding only;
- c) it uses, at most, two sets of Huffman tables per scan.

#### **I.6 The extended system**

The baseline system can be extended in many ways, one of them is to provide progressive coding of the DCT coefficients. In the progressive mode of operation a rough, but recognizable, version of the image appears quickly on the viewer's screen and is refined by successive passes to produce the same final image as the sequential mode. It is also extended to up to 12-bit source image, four sets of Huffman tables, arithmetic entropy coding option, etc.

The progressive coding modes are defined in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] by two complementary modes: successive approximation and spectral selection. For both progressive modes the stages prior to the Huffman encoding: DCT, quantization and coefficient ordering, are the same as for the sequential mode.

##### **I.6.1 Coding model for successive approximation**

In the successive approximation mode, the DCT coefficients are divided by an integer power of two before they are coded. The precision of the coefficients is therefore reduced. The coefficients are multiplied by the same power of two before the inverse DCT is applied. The precision of the coefficients is increased in the subsequent stages by reducing the scaling coefficient each time by a factor of two. In the final pass the DCT coefficients are coded with full precision. All coefficients, AC and DC, are scaled in the same way.

##### **I.6.2 Coding model for spectral selection**

In the spectral selection mode, the zigzag array of DCT coefficients described in Figure I.7 is segmented into “frequency” bands (called spectral bands) and each band is coded as a separate pass. In this progressive mode, the coding model of DCT coefficients is the same as for the sequential mode. However, the code tables of the baseline sequential system are extended to encode runs to end-of-band.

A particular use of the spectral selection mode is described in Appendix III.

### I.6.3 Hierarchical encoding

ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] defines another form of progressive coding: the hierarchical mode. The hierarchical mode provides a “pyramidal” encoding of an image at different resolutions, each resolution differing from the adjacent by a factor of two in either the horizontal or vertical direction, or both. The encoding procedure can be summarized as follows:

- 1) the original image is filtered and downsampled by the desired factor of  $2^N$  in each dimension;
- 2) the image of reduced size is encoded either sequentially or progressively according to the methods previously described;
- 3) the image of reduced size is upsampled by 2 in horizontal and/or vertical direction and used as the prediction for the next ( $2^{N-1}$ ) stage;
- 4) the  $2^{N-1}$  difference image is encoded.

This procedure is repeated until the final stage is reached. This capability is useful in applications where it is desirable to access a smaller version of an image or to support terminals with different resolutions from one database.

A more detailed description of one example of implementation of the hierarchical mode can be found in Appendix III.

## I.7 Summary

The algorithms developed by JPEG provide an excellent basis for the development of photovideotex services.

The baseline system ensures compatibility between simple systems using 8 bits per pixel per component. The sequential mode allows for cost effective implementations of the JPEG algorithm. The progressive system has advantages when a low speed network is being used. The image is built-up in stages and although the total build-up time is the same as with the sequential mode, the first scan of the image is displayed in less than the total time. The first scan of the image is often enough to allow the user to identify the image.

Within the different options provided by the JPEG algorithm, some useful selections have been made for “basic photovideotex” (compatible profiles in F.11), some are left for possible other selections for a private choice of photographic profile.

## I.8 Bibliography

- [I.1] WALLACE (G.): Overview of the JPEG still image compression standard, *Electronic imaging*, 1990, Boston.
- [I.2] JAIN (A.): Image data compression: A review, *Proceedings of the IEEE*, vol. 69, No. 3, March 1981.
- [I.3] JAIN (A.): Fundamentals of digital image processing, *Prentice Hall information and system sciences series*.
- [I.4] AHMED (N.), RAO (K.R.): Orthogonal transform for digital signal processing, *Springer Verlag*, New York.
- [I.5] AHMED (N.), NATARAJAN (T.), RAO (K.R.): Discrete cosine transform, *IEEE Trans. on Comput.*, January 1974.
- [I.6] HUFFMAN (D.A.): A method for the construction of minimum redundancy codes, *Proc. IRE*, vol. 40, September 1952.

**Appendix II**  
(Informative)  
(to Recommendation T.101, Annex F)

**Implementation guidelines on display rendering**

**II.1 Introduction**

Due to the fact that the three base Videotex syntaxes (ITU-T Data Syntax I-III) differ not only in their fundamental underlying parameters (such as: screen resolution, aspect ratio, terminal model, planes, order of planes, number of planes, etc.), but also in their terminal profiles (Data Syntax II), the photographic capability of this annex has been defined in a way which is independent of any base Videotex system.

This is necessary to ensure that any image can be displayed on any system, although certain artifacts might result if the display characteristics of the terminal differ from the transmitted source image. The process in which the terminal tries to cope with the mapping of the characteristics of the source image to the characteristics of its own display capability, is called rendering.

There are two types of rendering which are described in this appendix:

- rendering of resolution; and
- rendering of colour.

Since all rendering functions are local functions to the terminal (depending on the hardware and software capabilities of a given terminal configuration) and not part of the photographic data transfer, they are clearly outside of the scope of the normative part of this annex. However, because of the importance of these issues for terminal implementors some guidelines are provided to cope with rendering problems in the terminal.

**II.2 Rendering of resolution**

**II.2.1 Resolution independence**

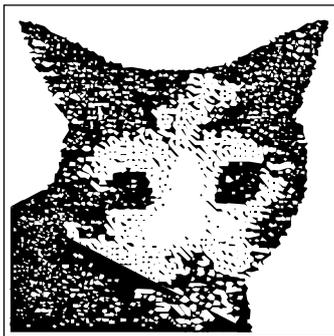
Independence of display resolution is an important factor even within a single Videotex system, since there may be a large number of different display platforms; such as various personal computer display boards or dedicated terminals. It is not desirable to restrict an information service to only certain groups, since this would restrict the market for the information provider.

In general, it is possible to display any array of pixels on any resolution display screen although certain presentation artifacts will result if the source and the target pixel array size do not match exactly. There are two ways of handling a mismatch between the source pixel array size and the target display pixel resolution. These are shrinking an image to fit or scaling the pixel array.

The following series of images show a photographic image which is rendered on a display device of different resolution, first by the shrinking technique and then by the scaling technique.

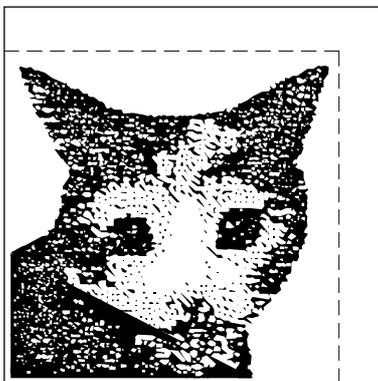
The shrinking technique results in an image which is somewhat smaller than intended by the information source; however, after the shrinking there results a simple ratio between the source (logical pixels) and the target (physical pixels). For example, if a source image was defined to occupy the full display area on a 255 by 300 pixel display screen and was rendered on a 240 by 320 display screen, then the image could be shrunk by 6.25% before display (see Figure II.2).

The other technique which can be used to render the display of a photographic image on a display screen of a different inherent resolution is to scale the pixel array. Logical pixels will be rendered onto the corresponding physical pixels of the target display system one by one across the display until the corresponding positional error is greater than a physical pixel position. The alignment of the source and the target pixel arrays would then be shifted over by one physical pixel and the rendering process would continue. This ensures that the relative size of the photographic image remains the same on target display devices. Effectively, this technique spreads the mismatch across the image. This is illustrated by Figure II.3.



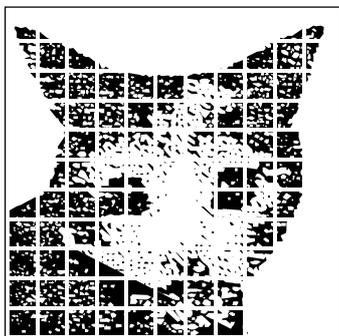
T0823280-95/d284

**Figure II.1/T.101 – Original image as intended by the information provider**



T0823290-95/d285

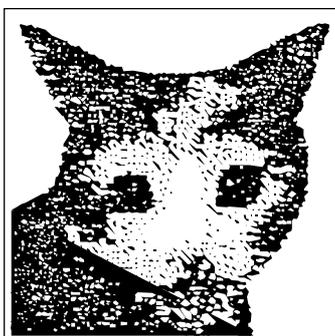
**Figure II.2/T.101 – Image rendered by shrinking technique**



T0823300-95/d286

**Figure II.3/T.101 – Image rendered by scaling technique (worst case approach – no averaging)**

The image above is a worst case situation. The gaps resulting from the scaling can be filled in by duplicating the previous pixel or by some more sophisticated averaging algorithm. The more resolution available in the luminosity or colour dimensions, then the better such an averaging technique works (see Figure II.4).



T0823310-95/d287

**Figure II.4/T.101 – Image rendered by scaling technique (averaging used to fill in gaps)**

In this annex any resolution is possible for both the target and the source display devices. Rendering of the image is the responsibility of the display device, scaling or shrinking of the image should be expected in some situations.

NOTE – Due to the possible effects of scaling or shrinking of a photographic image as a result of rendering, there is no guarantee that text, mosaic or geometric information will align with a photographic image exactly. It is not advisable to, for example, attempt to draw a moustache on a photographic image of a face or align text exactly with a photographic image. Such a restriction also helps to resolve the difference between the terminal models of the various Videotex systems, some of which ascribe separate underlying or overlaying display planes or a shared plane to the display of a photographic image. It is advisable to treat a photographic image separately and care should be taken when images are mixed with other drawing techniques.

## **II.2.2 Display rendering guidelines for Data Syntax II profiles**

### **II.2.2.1 Pixel alignment**

In principle the pixel resolution for the different layers (i.e. alpha-mosaic and photographic) are independent of each other, but in most implementations the pixel resolution should, most probably, be the same for all layers.

Since the shape of the characters are defined locally in the terminal the dot matrix size is only of importance for Dynamically Redefinable Character Sets (DRCS). At present, out of the four Videotex profiles of data syntax II defined in ITU-T Recommendation T.101 [1], only profile 1 includes DRCS.

A typical profile 1 dot matrix size is  $12 \times 10$  giving a defined display area resolution of  $480 \times 240$  or if the dots are grouped together  $2 \times 2$ ,  $960 \times 480$  in the alpha-mosaic layer. The dots of the DRCS could also be grouped in the vertical direction giving a defined display area resolution of only  $480 \times 480$ .

The following examples show how different resolutions of the photographic picture can be combined with profile 1 displays.

#### **II.2.2.1.1 CCIR Recommendation 601, Part 1: 4:2:2 resolution**

The *full visible display* resolution is  $720 \times 576$  pixels gives a *defined display area* resolution of  $640 \times 480$  pixels.

If a received  $640 \times 480$  image is to be displayed on a  $960 \times 480$  display and the aspect ratio should be kept (which it should), it is necessary to “add” 50% extra pixels horizontally. How that could be done is described in II.2.2.2.1.

The profile 1 terminal could also choose the  $480 \times 480$  resolution in which case 25% of the pixels should be “removed” horizontally from the received  $640 \times 480$  image. How that could be done is described in II.2.2.2.2.

### II.2.2.1.2 CCIR Recommendation 601, Part 1: 2:1:1 resolution

The *full visible display* resolution is  $360 \times 576$  pixels giving a *defined display area* resolution of  $320 \times 480$  pixels.

In this case a profile 1 terminal would most probably choose a resolution of  $480 \times 480$  giving the need for “adding” 50% pixels horizontally (II.2.2.2.1).

### II.2.2.1.3 CIF format

The *full visible display* resolution is  $360 \times 288$  pixels giving a *defined display area* resolution of  $320 \times 240$  pixels.

In this case a profile 1 terminal would most probably choose a resolution of  $480 \times 240$  giving the need for “adding” 50% pixels horizontally (subclause II.2.2.2.1).

### II.2.2.1.4 QCIF format

The *full visible display* resolution is  $180 \times 144$  pixels giving a *defined display area* resolution of  $160 \times 120$  pixels.

It is unlikely that this poor resolution will be chosen for Videotex terminals, but images in QCIF format can be sent to CIF terminals occupying one quarter of the screen.

## II.2.2.2 Adjustment of horizontal resolution

In some situations it is necessary to adjust the horizontal resolution to the resolution of the actual terminal. Some ways to do this are described below.

### II.2.2.2.1 “Adding” pixels

“Adding” 50% extra pixels (e.g. expansion of 640 pixels to 960 pixels) can be depicted as follows:

Pixel	A	B	C	in original picture	
	0	0	0		
	0	0	0	0	
Pixel	a	b	c	d	in expanded picture

1) *The simplest solution* is:

- a = A
- b = B
- c = B
- d = C

2) *Simple interpolation* gives:

- a = A
- b = (A + 2 \* B)/3
- c = (2 \* B + C)/3
- d = C

3) *Using a 2nd degree polynomial* gives:

$$\begin{aligned} a &= A \\ b &= (2 * A + 8 * B - C)/9 \\ c &= (-A + 8 * B + 2 * C)/9 \\ d &= C \end{aligned}$$

Also polynomials of higher degree can be used, but the small quality improvement observed between 1, 2 and 3 indicates that there is no extra benefit in doing this.

Subjective testing of the results of the proposed expansion methods shows that 1) gives visible artifacts where as artifacts from 2) and 3) can only be seen if the picture is enlarged by zooming. The conclusion is that 2) is sufficient for most applications while 1) can be used in terminals where computing power is limited.

#### II.2.2.2.2 “Removing” pixels

“Removing” 25% pixels (e.g. reducing 640 pixels to 480 pixels) can be depicted as follows:

Pixel	A	B	C	D	E	in original picture
	0	0	0	0	0	
	0	0	0	0		
Pixel	a	b	c	d		in reduced picture

1) *The simplest solution* is:

$$\begin{aligned} a &= A \\ b &= B \\ c &= D \\ d &= E \end{aligned}$$

2) *Simple interpolation* gives:

$$\begin{aligned} a &= A \\ b &= (2 * B + C)/3 \\ c &= (C + 2 * D)/3 \\ d &= E \end{aligned}$$

3) *Using a 2nd degree polynomial* gives:

$$\begin{aligned} a &= A \\ b &= (-A + 8 * B + 2 * C)/9 \\ c &= (2 * C + 8 * D - E)/9 \\ d &= E \end{aligned}$$

Also polynomials of higher degree can be used, but the small quality improvement observed between 1, 2 and 3 indicates that there is no extra benefit in doing this.

It is obvious that removing pixels results in loss of information and in the light of that 2) is sufficient for most applications while 1) can be used in terminals where computing power is limited.

#### II.2.3 Display rendering for Data Syntax III profiles

The Data Syntax III Videotex syntax is primarily an alpha-geometric syntax, and therefore it exhibits certain characteristics which ease display rendering of images of different resolution. The Defined Display Area (DDA) of a DS III terminal is specified as a resolution independent normalized coordinate space. All coordinates are specified as fractions of this normalized space and terminals typically have various resolutions. The base resolution of  $256 \times 200$  specified in the DS III Service Reference Model (SRM) is simply the minimum resolution at which the terminal should be constructed. Although a large number of low cost terminals are built to this resolution, a mix of terminal resolutions is used. Since a range of different terminal resolutions can be expected, the implementation of resolution independent rendering is important.

For DS III usage image rendering by the scaling technique is recommended. The pixel array communicated by the photographic data syntax can be converted for rendering to the pixel area of the target terminal by establishing a scaling factor which circumscribes the source picture within the target rendering area and then this scaling would be used to determine which pixels or lines must be doubled (or eliminated) in the rendering process. When normalized, the width of the original image in pixels, defines the logical width of a single picture element (Pel) as a binary fraction of the normalized area. The logical Pel establishes an x displacement across the width of the real display device upon which the image is rendered, which may be one or more physical pixels, but often is a fractional displacement (e.g. 1.25 real pixels per logical Pel). As long as the displacement is maintained to a precision greater than that required to represent the normalized width, to eliminate the accumulated round-off error, this process works in a completely resolution independent manner. A similar calculation takes place in the y direction. Optionally a running spatial average may be maintained so that pixel values which would need to be doubled up are averaged to ensure a smooth transition. The details of the algorithm required to support such rendering are given under the Incremental Point command in ITU-T Recommendation T.101, Annex D (DS III) [1].

### **II.3 The concept of normalised colour space**

The basic concept of a normalized colour space is that the photographic data syntax carries numbers from 0 (inclusive) to 1 (exclusive) which represents the range of the whole colour space.

If, for example, a byte of data is used to represent the colour, then 256 different colours can be represented. If, however, a terminal is unable to display 256 colours, the normalizing of the colour space allows the terminal to perform the “best fit mapping” of the available colour to the normalized value.

## **Appendix III**

(Informative)

(to Recommendation T.101, Annex F)

### **Solutions for common compatible photovideotex databases serving different resolution terminals**

#### **III.1 Introduction**

One of the major problems which has to be faced in the development of image services in general, and Videotex in particular, is the problem of a common database being accessed by terminals of different display resolutions.

In a Videotex database an image is always encoded and stored in the database prior to the user's access. Since many types of terminal are expected to be able to handle data from different databases, it is essential to provide interworking between a common database and the different resolution terminals, in order to provide:

- the opportunity for database providers to code images within the database with the aim that these images will be displayable on different resolution terminals;
- the opportunity for the users to choose their own terminal resolution with the aim that they have access to a broad range of images, but have some freedom in the selection of their display device.

To achieve this two solutions are offered:

- a) the “Hierarchical Mode” as defined in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13];
- b) “special Spectral selection”, a filtering technique in the transform domain, taking full advantages of the ITU-T Rec. T.81 | ISO/IEC 10918-1 [13] progressive coding algorithm (see Appendix VII).

### III.2 Hierarchical mode

In this technique, the basic idea is to use a sequence of downsampling filters to create a progression of reduced resolution images. Each downsampling reduces the resolution by a factor of two in either the horizontal or the vertical direction (or both).

The reduced resolution images are the input for each stage of the hierarchical progression.

#### III.2.1 Coding

As an example, Figure III.1 describes a three stages hierarchical DCT coding process giving output images with resolution: ITU-T Recommendation H.261 [6] QCIF and CIF and CCIR Recommendation 601 [9] 4:2:2.

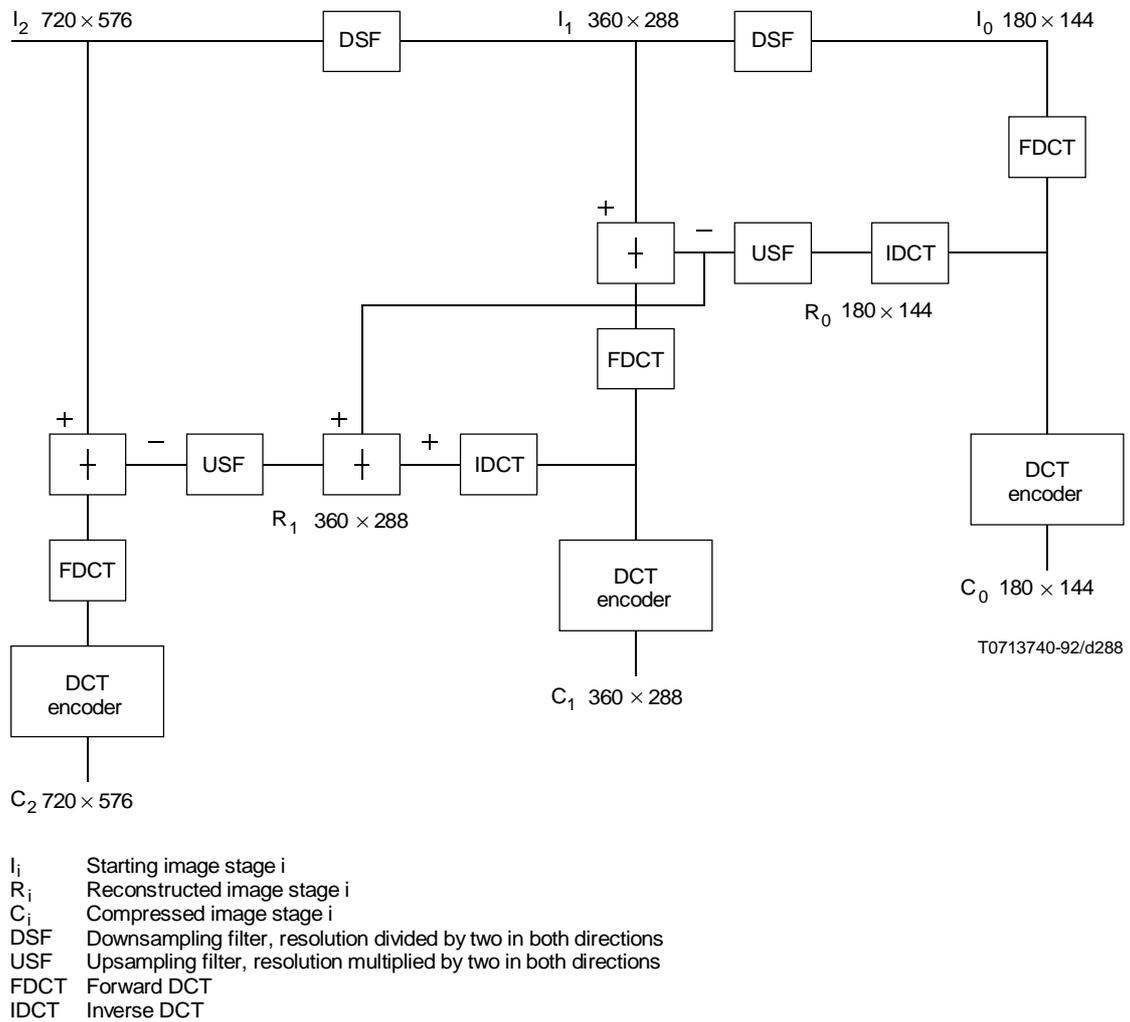


Figure III.1/T.101 – Hierarchical mode

Example of a three stages DCT coding process. Resolution is expressed as: number of pixels per line x number of lines, for the luminance.

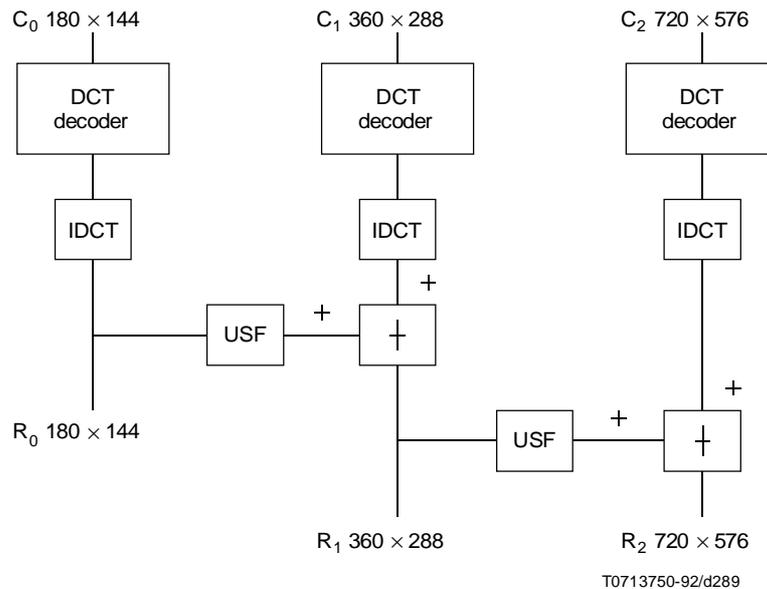
At the first stage of encoding (stage 0), the lowest resolution image  $I_0$  is coded using any of the compression algorithms defined in the ITU-T Rec. T.81 | ISO/IEC 10918-1 [13]: sequential, progressive, spatial, etc..

For further stages, say stage  $i$ , the reconstructed image from the previous stage,  $R_{i-1}$ , is upsampled and used as a prediction. The reconstructed image is obtained by summing the prediction and difference images. For these stages only difference images are coded giving compressed difference images,  $C_i$ , which can be stored or transmitted.

Each stage of progression provides an output compressed image whose resolution increases by a factor of two in both directions.

### III.2.2 Decoding

Figure III.2 gives the decoding scheme corresponding to the coding process described in Figure III.1.



**Figure III.2/T.101 – Hierarchical mode**

Example of a three stages DCT decoding process. All notations are the same as in Figure III.1.

At the receiving end, the lowest resolution compressed image,  $C_0$ , is first decoded in the same manner as for a non-hierarchical algorithm, giving the reconstructed image  $R_0$ .

At further stages of progression, say stage  $i$ , the previous reconstructed image,  $R_{i-1}$ , is upsampled and summed to the difference decoded image in order to obtain a reconstructed image,  $R_i$ , whose resolution is increased by a factor of two in both directions. So the receiver can stop to decode the incoming data stream when the desired resolution is reached.

The characteristics of the downsampling filter are left to the system designer but should be consistent with the upsampling bilinear interpolation filter, which is defined in the ITU-T Rec. T.81 | ISO/IEC 10918-1 [13].

### III.2.3 Example for a “resolution pyramid” for hierarchical build-up

In the example below the suggestion is to create a logical hierarchy of resolutions (both for luminance and chrominance) with appropriate spatial increases, covering the range from QCIF to HDTV.

The spatial increases start from a resolution which is obtained from the smallest size DC image (DC components of a CCIR Recommendation 601 [9] image). The spatial increase, horizontally and vertically, are decoupled and the single steps of resolution-scaling to the next higher level take the value of either 1 or 2, depending on a logical, strictly defined scaling strategy. The strategy is such, that during the scaling process to a higher resolution the “main stages” shown in Table III.1 are covered.

**Table III.1/T.101 – Main stages in hierarchical progression**

Main stages	
A	DC only
B	Recommendation ITU-T H.261 [6]: QCIF
C	Recommendation ITU-T H.261 [6]: CIF
D	Recommendation CCIR 601 [9]: 2:1:1
E	Recommendation CCIR 601 [9]: 4:2:2
F	HDTV 4:2:2

This actually is achieved as shown in Table III.2.

For example, to get to Stage F, 9 progressive stages would be needed. To get to stage C, 4 progressive stages would be needed.

**Table III.2/T.101 – Details of hierarchical progression**

Main Stage/ Progressive Step	Luminance				Chrominance			
	Resolution				Resolution			
	Horizontal	Scale	Vertical	Scale	Horizontal	Scale	Vertical	Scale
A/1	90	2	72	2	90	1	72	1
B/2	180	1	144	2	90	1	72	2
/3	180	2	288	1	90	2	144	1
C/4	360	1	288	1	180	1	144	2
/5	360	1	288	2	180	1	288	2
D/6	360	2	576	1	180	2	576	1
E/7	720	1	576	2	360	1	576	2
/8	720	2	1152	1	360	2	1152	1
F/9	1420	–	1152	–	720	–	1152	–

### III.2.3.1 Advantages of the suggested technique

#### III.2.3.1.1 Independence of transmission, decompression and display

It is up to the terminal to decide which progressive stage to display on the screen and how. For example, on a high speed network, the terminal may decide to show only the DC-scan, then leave out all progressive stages until CCIR 601 [9] 2:1:1, and to display that as the final resolution. On a lower speed network, the terminal could show all intermediate stages, in order to achieve a more pleasing image build-up.

Since progressive image build-up requires a frame buffer, the speed of transmission, image decompression, and image display might be entirely decoupled from one another (for the terminal to decide). Thus, even if the network speed is high, the decompression and display could be done at lower speeds (the progressive build-up is more tolerant to variations of speed). This could be the case if a single database is to serve terminals linked through both high- and low-speed channels (thus only progressive image build-up is practical).

### III.2.3.1.2 Independence of image resolution from the terminal resolution

In the rendering process the adjustment of the output display seems to be easier if there are “fixed” progression stages, where the image adjustment can be fixed (e.g. through interpolation between two stages, or by simple selection of a given stage).

### III.2.3.1.3 Storage gain in the database host through the pyramidal database

In hierarchical mode the information provider can select the final resolution (e.g. CCIR 601 [9] 4:2:2) for each stored image separately. In this respect the “special spectral selection” technique is more rigid, because, as an analogy, it would require that all images be stored according to the highest resolution to be displayed (e.g. CCIR 601 [9] 4:2:2).

### III.2.3.2 Disadvantages

The hierarchical mode, elegant in principle, suffers from one drawback: its implementation complexity.

Compared to the non-hierarchical coding techniques, the hierarchical mode requires more calculation power and memory :

- In addition to computation of forward DCT, the encoder has to perform: downsampling and upsampling filtering, inverse DCT computation and difference computation.

Additional memory is also required to store the upsampled image prior to difference computation.

- In addition to computation of inverse DCT, the decoder has to perform bilinear interpolation filtering.

Additional memory is required to store the upsampled reconstructed image prior to accumulation with the decoded difference image (plus extra precision bits for accumulation).

- Reference images may diverge due to IDCT differences.
- Substantially more bits are needed to achieve the final stage.

## III.3 Special spectral selection

In the hierarchical mode, downsampling and upsampling filtering are used at both coding and decoding levels to obtain different resolution images.

The technique described below is based on the following idea:

- the image is coded with only one given resolution (basic resolution, typically “CCIR 601 [9] 4:2:2”);
- at the receiving end, instead of performing a “traditional”  $8 \times 8$  inverse DCT followed by a downsampling filtering of the image to obtain the desired resolution, the image is decoded using a variable size inverse DCT.

This sort of “DCT filtering” allows images with different resolutions to be displayed without having to perform any postfiltering. Variable size inverse DCTs are performed by dropping out “the unused” DCT coefficients (see Figure III.3).

This technique can be applied to any of the DCT coding algorithms defined in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13], but due to the variable size inverse DCT, it is more efficient when used in combination with the progressive coding algorithm, as shown below.

NOTE – See also Appendix VII (informative) to this Annex.

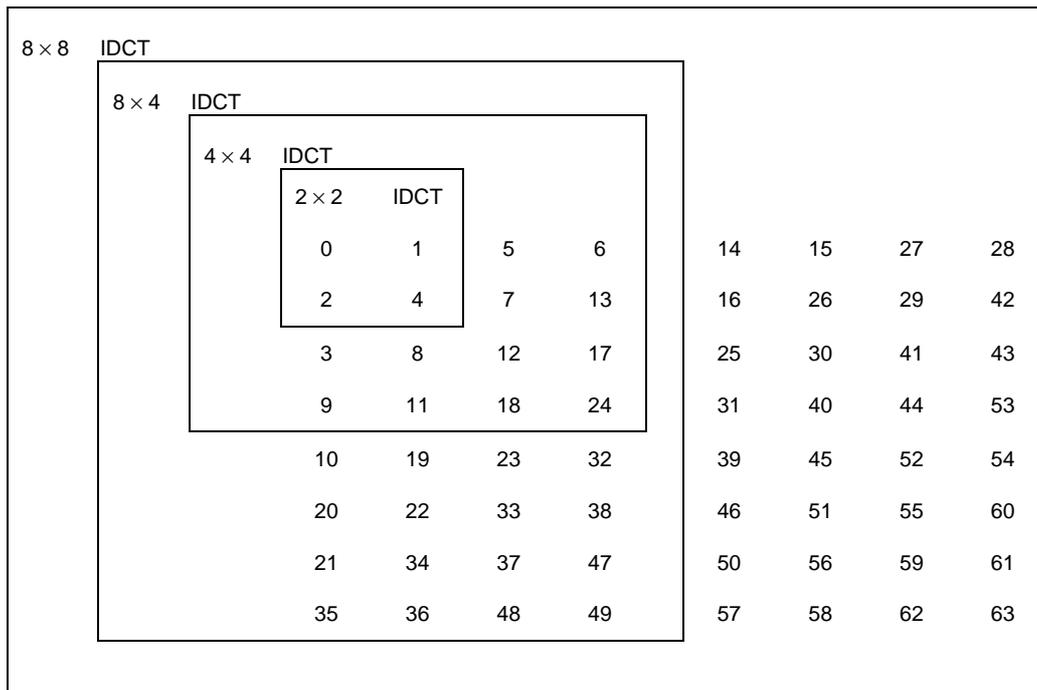


FIGURE III.3/T.101

**DCT coefficients required to perform the different size inverse DCT used to display a CCIR 601 [9] 4:2:2 coded image with QCIF, CIF, CCIR 601 [9] 2:1:1 and CCIR 601 [9] 4:2:2 resolutions**

### III.3.1 Coding

When considering the encoding progressive DCT algorithm (described in ITU-T Rec. T.81 | ISO/IEC 10918-1 [13]), a specific definition of the spectral bands can be used in order to improve the efficiency of the technique. See also I.6.2.

The benefits of “well specified” spectral bands are that the receiver will only have to decode the spectral bands containing the DCT coefficients required to perform a given size inverse DCT.

Figure III.3 shows the DCT coefficients required to perform the different size inverse DCT used to display a 4:2:2 coded image with QCIF, CIF, 2:1:1 and 4:2:2 resolutions.

According to the display resolutions mentioned above, a CCIR 601 [9] 4:2:2 image can then be coded with the following spectral bands:

- first spectral band: coefficient 0 (DC coefficient only, since mixing of DC and AC coefficients in the same spectral band is not allowed according to ITU-T Rec. T.81 | ISO/IEC 10918-1 [13]);
- second spectral band: from coefficients 1 to 4;
- third spectral band: from coefficients 5 to 24;
- fourth spectral band: from coefficients 25 to 49;
- fifth spectral band: from coefficients 50 to 63.

Considering these proposed spectral bands, the image is coded using the “classical” progressive coding algorithm defined by ITU-T Rec. T.81 | ISO/IEC 10918-1 [13].

This definition of the spectral bands is not mandatory, other spectral bands can be defined. Spectral bands can be added (within each spectral band defined above) in order to improve the progressive aspect of the display. As another example, the third spectral band can be reduced from coefficients 1 to 13 in order to reduce from 9 to 1 the number of extra coefficients included in the spectral band due to the zigzag scanning of DCT coefficients. In this case, the transmission time of the three first spectral bands is noticeably reduced with only minor degradations of the image quality (due to the absence of AC coefficients 17, 18 and 24).

### III.3.2 Decoding

At the decoding level, the size of the inverse DCT is adjusted to fit with the desired image resolution. The inverse DCT calculation involves only the required DCT coefficients, as described in Figure III.3. As a result, only relevant spectral bands have to be decoded.

Extra coefficients included in the spectral bands, due to the zigzag scanning of the DCT coefficients, are dropped out by the decoder before performing the inverse DCT.

The display of a CCIR 601 [9] 4:2:2 coded image with the resolution mentioned above required the following processing:

- *icons:*  
first spectral band decoded (DC coefficient)  
 $1 \times 1$  inverse DCT;
- *180 pixels  $\times$  144 lines resolution (QCIF format):*  
two first spectral bands decoded  
 $2 \times 2$  inverse DCT;
- *360 pixels  $\times$  288 lines resolution (CIF format):*  
three first spectral bands decoded  
 $4 \times 4$  inverse DCT;
- *360 pixels  $\times$  576 lines resolution (2:1:1 format):*  
four first spectral bands decoded  
 $8 \times 4$  inverse DCT;
- *720 pixels  $\times$  576 lines resolution (4:2:2 format):*  
all the spectral bands decoded  
 $8 \times 8$  inverse DCT;

Other processing can be envisaged. For example, a CIF resolution image can be obtained by decoding only the two first spectral bands and by performing a  $4 \times 4$  inverse DCT on these coefficients. This produces a reasonable image in terms of quality and resolution for a very low transmission and computation cost.

In order to take full benefit of the use of spectral bands, the inverse DCT can be calculated using an accumulation technique on the elementary sub-images corresponding to each DCT coefficient (depending on the number of coefficients to be processed in each spectral band).

Compared to the hierarchical mode, the main advantage of this technique is its simplicity.

- The coding technique is a “classical”, non-hierarchical technique. So no increase in the calculation power is required and also no additional memory is required.
- The complexity of the decoder is adapted to the resolution of the terminal. Depending on its resolution, the terminal only decodes the necessary spectral bands and performs an adjusted size inverse DCT. So low resolution terminals only have to perform a simple inverse DCT (typically  $2 \times 2$  or  $4 \times 4$  inverse DCT for a 4:2:2 basic resolution).
- In addition, no post-filtering is required in the decoder.

Since the number of inverse DCTs with different sizes is limited, the range of resolution covered by this technique is also limited. The lowest available resolution (corresponding to  $1 \times 1$  inverse DCT) is 1/64 of the upper resolution (basic resolution). However, 4:2:2, 2:1:1, CIF and QCIF resolutions can be handled from within the same image.

**Appendix IV**  
(Informative)  
(to Recommendation T.101, Annex F)

**Coding examples**

**IV.1 Introduction**

This informative appendix provides examples for coding according to the syntax defined by this annex.

**IV.1.1 Example 1**

In this example the simplest way of transmitting a photographic image is assumed, using as many default values of attributes, parameters and tables as possible.

The transmission sequence of two consecutive, but separate photographic frames is coded. Both Videotex frames include photographic information only and for simplicity no alpha-mosaic or geometric data. The photographic image of both frames should fill the entire DDA. The compressed pictures each have a size of 10 000 bytes. The picture data is sent in blocks with a maximum size of 4000 bytes, which allows an interruption in the transmission if required.

*Picture 1:*

**ISO 9281**

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 4/11>

**Header information**

<PDE<sub>1</sub>>

<5/1>

Reset to Default

<RTD> <dev>

<2/0 3/0> <4/5 0/1 0/1>

Clear Photo-area

<CPA> <cle>

<2/1 3/5> <4/5 0/1 0/1>

**ISO 9281**

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/3 7/13 4/1>

<PDE<sub>1</sub>>

<5/2>

(4000 bytes of Image Data)

**ISO 9281**

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/3 7/13 4/1>

<PDE<sub>1</sub>>

<5/2>

(4000 bytes of Image Data)

**ISO 9281**

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/1 7/14 5/1>  
<PDE<sub>1</sub>>

<5/3>

(2000 bytes of Image Data)

*Picture 2:*

**ISO 9281**

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 4/6>

**Header information**

<PDE<sub>1</sub>>

<5/1>

Clear photo -area

<CPA> <cle>

<2/1 3/5> <4/5 0/1 0/1>

**ISO 9281**

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/3 7/13 4/1>

<PDE<sub>1</sub>>

<5/2>

(4000 bytes of Image Data)

**ISO 9281**

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/3 7/13 4/1>

<PDE<sub>1</sub>>

<5/2>

(4000 bytes of Image Data)

**ISO 9281**

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/1 7/14 5/1>

<PDE<sub>1</sub>>

<5/3>

(2000 bytes of Image Data)

**IV.1.2 Example 2**

In this example a more complex way of transmitting a photographic image is assumed, using some customized tables.

The transmission sequence of two consecutive, but separate photographic frames is coded. Both Videotex frames should include photographic information only and for simplicity no alpha-mosaic or geometric data. The photographic image of both frames should fill the entire DDA. Frame 1 should include downloading of one custom quantization table and two sets of custom Huffman tables. A set of Huffman tables includes both AC and DC Huffman tables for a given image component. A separate PE will be used for downloading the new custom tables. Frame 2 should not include the downloading of any tables, but uses the tables downloaded with frame 1.

The compressed pictures each have a size of 10 000 bytes. The picture data is sent in blocks with a maximum size of 4000 bytes, which gives a shorter delay if an interruption in the transmission is required.

In this example a mixture of custom and default tables are used. The parameter reset to default instructs the application to load all the default tables, subsequently the encoding table management parameter informs the application that some custom tables will be sent. The length of the tables is assumed to be 2000 bytes.

For the second picture the encoding table management parameter is used to inform the application that the “in use” tables are still valid.

*Picture 1:*

### ISO 9281

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/1 4/12>

### Header information

(44 bytes of Photographic Header Data)

<PDE<sub>1</sub>>

<5/1>

Reset to Default

<RTD> <dev>

<2/0 3/0> <4/5 0/1 0/1>

Clear Photo-area

<CPA> <cle>

<2/1 3/5> <4/5 0/1 0/1>

Encoding table management

<ETM> <ttp> <tid> <tst> <ETM> <ttp> <tid> <tst> <ETM> <ttp> <tid> <tst>

<2/4 3/1> <4/4 0/1 0/1> <4/0 0/1 0/7> <4/4 0/1 0/3>

<2/4 3/1> <4/4 0/1 0/2> <4/0 0/1 5/5> <4/4 0/1 0/3>

<2/4 3/1> <4/4 0/1 0/2> <4/0 0/1 5/6> <4/4 0/1 0/3>

### ISO 9281

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/3 7/13 4/1>

<PDE<sub>1</sub>>

<5/2>

[4000 bytes of Photographic Data (with tables)]

### ISO 9281

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/3 7/13 4/1>

<PDE<sub>1</sub>>

<5/2>

(4000 bytes of Photographic Data)

### ISO 9281

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/3 7/13 4/1>

<PDE<sub>1</sub>>

<5/2>

(4000 bytes of Photographic Data)

**ISO 9281**

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/1 7/14 5/1>

<PDE<sub>1</sub>>

<5/3>

(2000 bytes of Photographic Data)

*Picture 2:*

**ISO 9281**

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/1 4/7>

**Header information**

(39 bytes of Photographic Header Data)

<PDE<sub>1</sub>>

<5/1>

Clear Photo-area

<CPA> <cle>

<2/1 3/5> <4/5 0/1 0/1>

Source table specification

<ETM> <ttp> <tid> <tst> <ETM> <ttp> <tid> <tst> <ETM> <ttp> <tid> <tst>

<2/4 3/1> <4/4 0/1 0/1> <4/0 0/1 0/7> <4/4 0/1 0/2>

<2/4 3/1> <4/4 0/1 0/2> <4/0 0/1 5/5> <4/4 0/1 0/2>

<2/4 3/1> <4/4 0/1 0/2> <4/0 0/1 5/6> <4/4 0/1 0/2>

**ISO 9281**

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/3 7/13 4/1>

<PDE<sub>1</sub>>

<5/2>

(4000 bytes of Image Data)

**ISO 9281**

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/3 7/13 4/1>

<PDE<sub>1</sub>>

<5/2>

(4000 bytes of Image Data)

**ISO 9281**

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/3 7/14 5/1>

<PDE<sub>1</sub>>

<5/3>

(2000 bytes of Image Data)

### IV.1.3 Example 3

In this example a single more complex Videotex frame including alpha-mosaic and photographic data will be coded. First, the alpha-mosaic part of the frame is transmitted, then by switching to the photographic mode a photo-area is defined where a baseline JPEG image according to CCIR 601 [9] 4:2:2 will be placed. The placement into the photo-area should be according to the default values. The compressed pictures each have a size of 10 000 bytes. The picture data is sent in blocks with a maximum size of 4000 bytes, which gives a shorter delay if an interruption in the transmission is required.

#### Alpha-mosaic:

<US> <TEXT and Control codes>

<1/14> <4/1 4/13 2/0 7/4 6/5 7/3 7/4 0/10 0/13> "AM test"

*Picture:*

#### ISO 9281

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 5/10>

#### Header information

(26 bytes of header information)

<PDE<sub>1</sub>>

<5/1>

Reset to Default

<RTD> <dev>

<2/0 3/0> <4/5 0/1 0/1>

Clear Photo-area

<CPA> <cle>

<2/1 3/5> <4/5 0/1 0/1>

"TRUE"

Source Picture Dimension.

<PDS> <nph> <npv>

<2/2 3/1> <4/0 0/2 0/5 0/0> <4/0 0/2 0/3 6/0>

"640 × 480 pixel"

Source Pixel Density.

<PID> <stf>

<2/2 3/2> <4/4 0/1 0/1>

"4:2:2 625 lines"

#### ISO 9281

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/3 7/13 4/1>

<PDE<sub>1</sub>>

<5/2>

(4000 bytes of Image Data)

#### ISO 9281

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/3 7/13 4/1>

<PDE<sub>1</sub>>

<5/2>

(4000 bytes of Image Data)

## ISO 9281

<PCD> <PM> <PI> <LI>

<1/11 7/0> <2/3> <4/0> <7/15 6/1 7/14 5/1>  
<PDE<sub>1</sub>>

<5/3>

(2000 bytes of Image Data)

### IV.2 Image positioning examples

In the examples only those sub-parameters needed to position the picture on the screen are included.

#### IV.2.1 Example 1: 640 × 480 picture inside DDA

<FSD> <ful> = "false"

<ASR> <arah> = 4

<araw> = 3

<LOC> <loch> = 0

<locv> = 0

<PAS> <sizw> = 1

<sizh> = 0.75

<PPL> *either (lower left corner ref.):*

<refh> = 0

<refv> = 479

<offh> = 0

<offv> = 0

*or (upper left corner ref.):*

<refh> = 0

<refv> = 0

<offh> = 0

<offv> = 0.75

*or:*

default

<PDS> <nph> = 640

<npv> = 480

<PID> <stf> = 4:2:2, 625 líneas

#### IV.2.2 Example 2: 720 × 576 picture full screen

<FSD> <ful> = "true"

<ASR> <araw> = 5

<arah> = 4

<LOC> <loch> = 0

<locv> = 0

<PAS> <sizw> = 1

<sizh> = 0.8

<PPL> *either (lower left corner ref.):*

<refh> = 0

<refv> = 575

<offh> = 0

<offv> = 0

*or (upper left corner ref.):*

<refh> = 0  
<refv> = 0  
<offh> = 0  
<offv> = 0.8

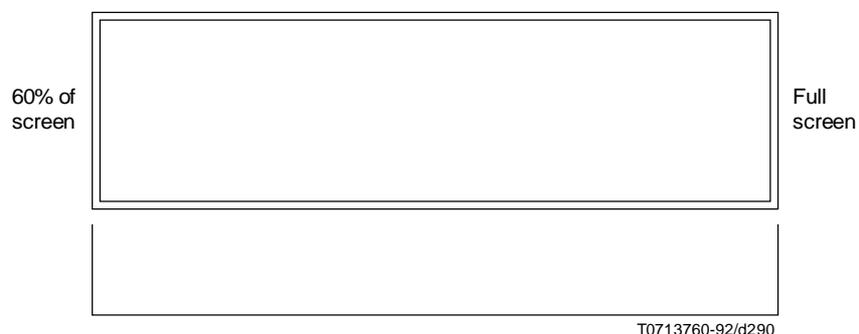
*or:*

default

<PDS> <nph> = 720  
          <npv> = 576  
<PID> <stf> = 4:2:2, 625 lines

### IV.2.3 Example 3: 720 × 346 picture covering upper 60% of full screen

See Figure IV.1.



T0713760-92/d290

**Figure IV.1/T.101 – 60% of the screen is filled with a photographic image**

<FS>           <ful> = "true"

<ASR>           <araw> = 5  
                  <arah> = 4

<LOC>           <loch> = 0  
                  <locv> = 0.32

<PAS> <sizw> = 1  
          <sizh> = 0.48

<PPL> *either (lower left corner ref.):*

<refh> = 0  
<refv> = 345  
<offh> = 0  
<offv> = 0

*or (upper left corner ref.):*

<refh> = 0  
<refv> = 0  
<offh> = 0  
<offv> = 0.48

<PDS> <nph> = 720  
<npv> = 346

<PID> <stf> = 4:2:2, 625 lines

### IV.3 Example for Source Picture Comments (PCT)

In the following a fictitious example of a possible use of the Source Picture Comment (PCT) is provided.

#### IV.3.1 An application scenario

Assume that the photographic picture coded according to this Recommendation of the “Mona Lisa” should be decorating the upper right corner of a Videotex frame describing the services of the “Beauty Salon Butterfly”. Assume that no text covering the “Mona Lisa” as a painting (or as a source image) is displayed in Videotex Alpha-Mosaic mode, since this would be disturbing for the main text message of the frame. Also assume that “Studio Alpha”, however, which sells this picture to several information providers (including the “Paris Tourist Board”, the “Leonardo da Vinci Foundation of Homeless Painters”) insist, as the picture source provider, to put, as a general policy of the Studio, in all applications a source picture comments linked to the “Mona Lisa” picture, for copyright and reference purposes.

The description of the logical record structure is provided by the source picture provider. If the use of the PCT parameter is permitted by the Photographic profile, then the actual use of PCT is left to the retrieving terminal. Thus, one class of terminals may simply recognize the code of the PCT parameter, and those terminals may simply wish to skip the number of bytes that is given for the length of the picture comment. Another class of terminals, which are for example linked to a small local picture data base, where images get downloaded to and from the central Videotex system, may wish to use the content of the picture comments as additional keywords.

For example, in the above case, the Studio might wish to know whether their source picture was incorporated into the Videotex frame of the beauty salon or not, and, if so, which version of their “Mona Lisa” was used. Further, the “Paris Tourist Board” may download and use the “Mona Lisa” in their own local data base.

The use of this parameter is intended to be extremely flexible.

#### IV.3.2 Sample logical record of a source picture

Assume that the logical sequence of the fields in the record are fixed. Since the field-length is variable, each field is preceded by a length indicator, which is 1 byte long (thus each field can be maximally 256 bytes long, assuming binary coding for the length). Assume the following logical record:

- Picture reference number (0): 123456
- Title of source (1): “Mona Lisa”
- Painter (2): Leonardo da Vinci
- Painted (3): 1502-1506
- Classification (4): Italian master
- Location (5): Louvre, Paris, France
- Reproduction date (6): 15/12/1995
- Photographer (8): X.Y.
- Copyright photo (9): Studio Alpha, Sollnerstr 10, 8000 München 70, Germany

Assume the following physical coding according to 7-bit ITU-T Recommendation T.51 [7], where it is assumed that no control characters (except for code extension according to ISO 2022 [10]) are needed to “process” this comment by the receiving terminal.

**Table IV.1/T.101 – Example of source picture comments coding**

Field No. (i)	Text (i)	Length code (i)	ITU-T T.51 Text Code (i)
0	123456	X '06'	3/1 3/2 3/3 3/4 3/5 3/6
1	Mona Lisa	X '09'	4/13 6/15 5/14 6/1 2/0 4/12 6/9 7/3 6/2
2	Leonardo da Vinci	X '11'	4/12 6/5 6/15 6/14 6/1 7/2 6/4 6/15 2/0 6/4 6/2 2/0 5/6 6/9 6/14 6/3 6/9
etc.			
9	Studio Alpha, Sollnerstr. 10, 8000 München 70, Germany	X '35'	5/3 7/4 6/4 6/9 6/15 2/0 4/1 6/12 7/0 6/8 6/1 2/0 2/12 2/0 5/3 6/15 6/12 6/12 6/14 6/5 7/2 2/14 2/0 3/1 3/0 2/12 2/0 3/8 3/0 3/0 3/0 2/0 4/13 1/11 6/14 4/8 0/15 7/5 6/14 6/3 6/8 6/5 6/14 2/0 3/7 3/0 2/12 2/0 4/7 6/5 7/2 6/13 6/1 6/14 7/9

Then the picture comment character string is the following:

<PCT> <cmt>

whereby: <PCT> = 2/1 3/0

<cle> = ...7-bit code...

<cid> = ...T.51...

In the JPEG Data stream: search for the comment (COM) marker-code (x'FFFE'):

<length> <value> (string)

whereby <length> = Length of the "comment character string" (150 bytes)

= 04/02 01/06

<value> (string) = <Length code (0)> <Text code (0)>

<Length code (1)> <Text code (1)>

<Length code (9)> <Text code (9)>

NOTE – The COM string in the JPEG assumes 8-bit "byte", though the text string was coded for transmission in the 7-bit "in-use" form of ITU-T Recommendation T.51 [7]. Thus, the most significant bit of each JPEG COM byte remains insignificant.

## Appendix V

(Normative)

(to Recommendation T.101, Annex F)

### Encoding parameters values for the 2:1:1 derived from CCIR Recommendation 601, Part 1 [9]

#### V.1 Introduction

Due to the fact that one of the compatible profiles defined in the F.11 has a resolution based on the "2:1:1" format and that furthermore CCIR Recommendation 601 [9] does not fully define the 2:1:1 format, it is necessary to give in this appendix, the technical specifications.

Also, this is absolutely necessary to ensure that any 2:1:1 image can be displayed on any system, to avoid annoying artifacts if the display characteristics of the terminal differ from the transmitted source image. The process in which the terminal tries to cope the mapping of the characteristics of the source image to the characteristics of its own display capability, is treated in Appendix II (display rendering guidelines).

The technical specifications of the 2:1:1 is based on the general principles outlined in CCIR Recommendation 601 [9] and is directly derived from the parameters defined for 4:4:4 and 4:2:2. So, the needed complements to the CCIR Recommendation 601 [9] for a definition of the 2:1:1 format are given in this appendix.

## **V.2 Encoding parameters for 2:1:1**

### **V.2.1 Main body of CCIR Recommendation 601, Part 1 [9]**

Paragraphs 1, 2 and 3 are fully applicable to the new set of parameters.

Paragraph 4 defines the encoding parameter values for the 4:2:2 member. This is still valid as a reference for the derivation of the 2:1:1 member. This leads to the definition of a new set of values as outlined in Table V.1.

NOTE: The derivation is straightforward for the 625/50 system. However, for the 525/60 system some precautions should be taken (see Table V.1).

### **V.2.2 Appendix I of the CCIR Recommendation 601, Part 1 [9]**

The tentative specification of the 4:4:4 member is still valid as a reference in the CCIR 601 compatible family.

### **V.2.3 Appendix II of the CCIR Recommendation 601, Part 1 [9]**

#### **V.2.3.1 Relationship of active line to analogue synchronization reference**

The relationship between 360 digital active line luminance samples and the analogue synchronizing references for the 625-line systems and the 525-line systems are shown in Table V.2.

The respective numbers of colour-difference samples can be obtained by dividing the number of luminance samples by two. The (6, 66) and (8, 61) were chosen symmetrically to position the digital active line within the permitted variations. They do not form part of the digital line specification and relate only to the analogue interface.

#### **V.2.3.2 Definition of the digital signals $Y$ , $C_B$ , $C_R$ , from the primary (analogue) signals $E'_R$ , $E'_G$ and $E'_B$**

The definition applied to the 4:2:2 shall be fully applicable to the 2:1:1.

### **V.2.4 Appendix III of the CCIR Recommendation 601 [9]**

The filtering characteristics are modified according to tables V.3, V.4 and V.5 of this appendix.

They are directly derived from the 4:2:2 filtering specifications.

NOTE – The passband ripple tolerance and the passband group-delay tolerance are specified in full respect to CCIR Recommendation 601 [9], but they might be found too stringent to certain applications. It is felt that the tolerances should be made more flexible for the 2:1:1. This flexibility should be specified. It is proposed in a first stage to permit the doubling of the tolerances on the passband ripple and group-delay.

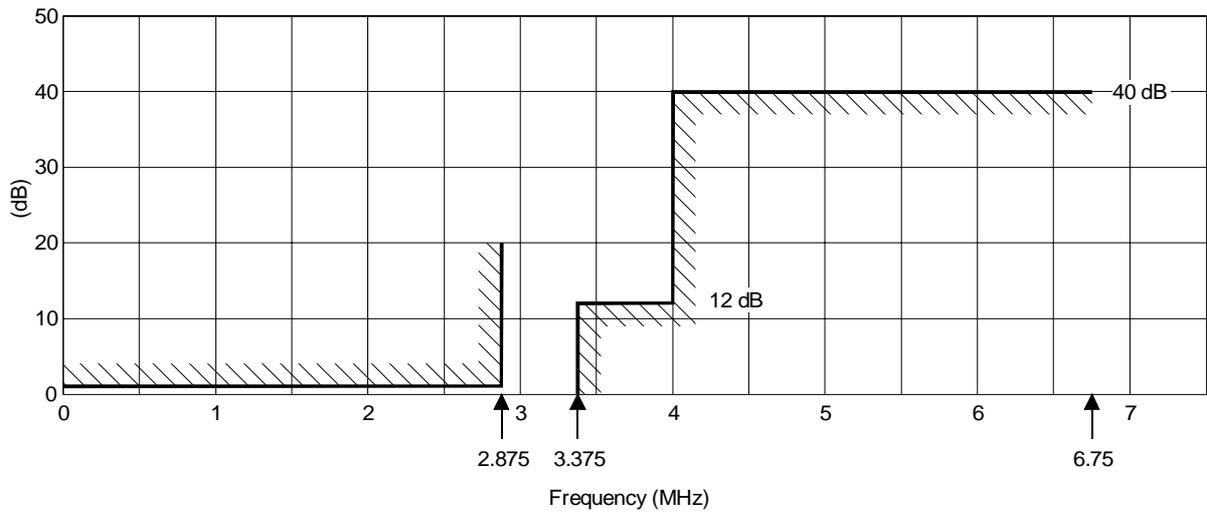
**Table V.1/T.101 – Encoding parameter values for the 2:1:1 member of the family**

Parameters	525-line, 60 field/s systems (Note 1)	625-line, 50 field/s systems (Note 1)
1. Coded signals: Y, C <sub>B</sub> , C <sub>R</sub>	These signals are obtained from gamma pre-corrected signals, namely: E' <sub>Y</sub> , E' <sub>R</sub> - E' <sub>Y</sub> , E' <sub>B</sub> - E' <sub>Y</sub> (Annex II, clause 2 refers in Recommendation 601)	
2. Number of samples per total line: – luminance signal (Y) – each colour-difference signal (C <sub>B</sub> , C <sub>R</sub> )	429 215 (Note 2)	432 216
3. Sampling structure	Orthogonal, line, field and frame repetitive, C <sub>B</sub> and C <sub>R</sub> samples co-cited with even (2nd, 4th, ...) odd (1st, 3rd, ...) Y samples in each line (Note 3)	
4. Sampling frequency: – luminance signal – each colour-difference signal	6.75 MHz (Note 4) 3.375 MHz  The tolerance for sampling frequencies should coincide with the tolerance for the line frequency of the relevant colour television standard	
5. Form of coding	Uniformly quantized PCM, 8 bits per sample, for the luminance signal and each colour-difference signal	
6. Number of samples per digital active line: – luminance signal – each colour-difference signal	360 180	
7. Analogue-to-digital horizontal timing relationship: – from end of digital active line to 0 <sub>H</sub>	8 luminance clock periods	6 luminance clock periods
8. Correspondance between video signal levels and quantization levels: – scale – luminance signal – each colour-difference signal	0 to 255 220 quantization levels with the black level corresponding to level 16 and the peak white level 235. The signal level may occasionally excursion beyond level 235 225 quantization levels in the centre part of the quantization scale with zero signal corresponding to level 128	
9. Code-word usage	Code-words corresponding to quantization levels 0 and 255 are used exclusively for synchronization. Levels 1 to 254 are available for video	
NOTES		
1 See CCIR report 624 (Characteristics of TV systems), Table 1.		
2 In 525/60 system the odd number of luminance samples per total line means a non-integer number of colour difference samples.		
3 In 525/60 system, the respect of item 3 implies half period offset of colour-difference sampling clock every line, in order to get co-sited samples (C <sub>B</sub> , C <sub>R</sub> and Y) with the beginning of the digital active line.		
4 The sampling frequency of 6.75 MHz (luminance) is an integer multiple of 2.25 MHz, the lowest common multiple of the line frequencies in 525/60 and 625/50 systems.		

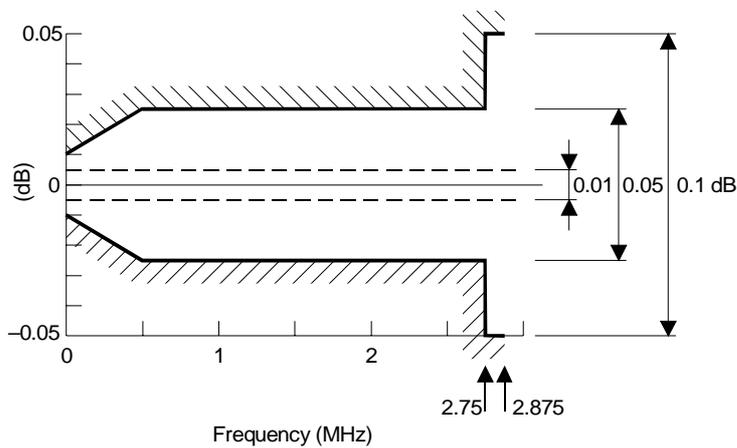
**Table V.2/T.101 – Relationship of digital active line to analogue synchronization reference**

525-line, 60 fields/s systems	$61 T$	$360 T$	$8 T$	
$0_H$ (Leading edge of line syncs., half-amplitude reference)		Digital active-line period		Next line $0_H$
625-line, 50 field/s systems	$66 T$	$360 T$	$6 T$	
$T$ One luminance sampling clock period (128 ns nominal).				

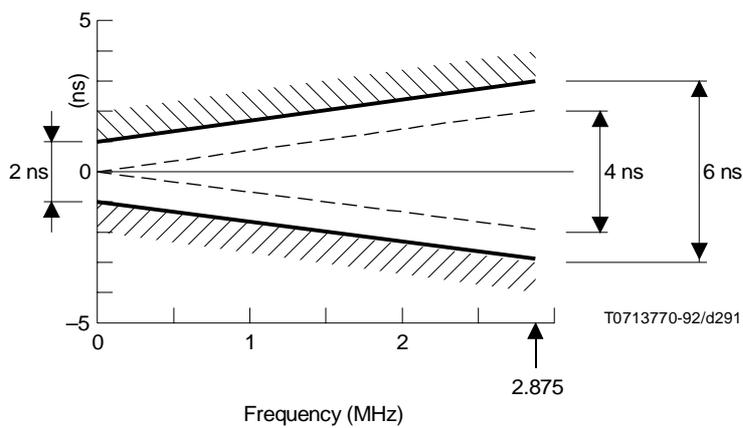
**Table V.3/T.101 – Specification for a luminance or RGB signal filter used when sampling at 6.75 MHz**



**a) Template for insertion loss/frequency characteristic**



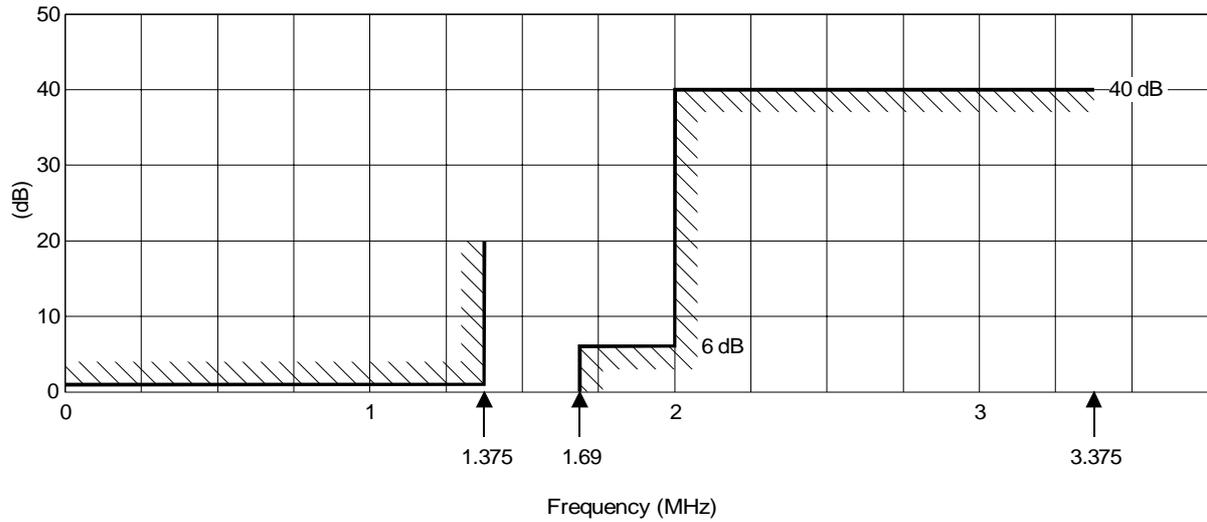
**b) Passband ripple tolerance**



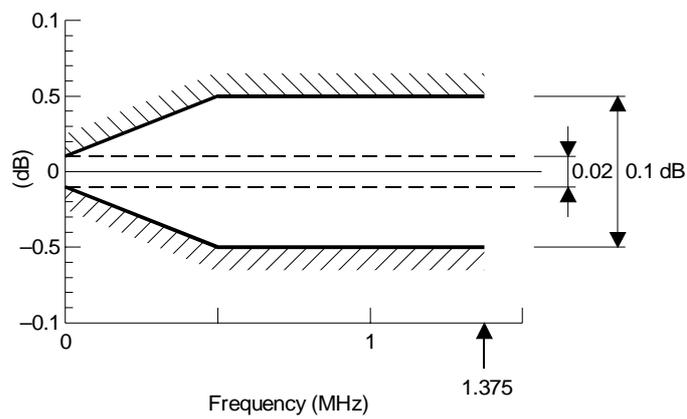
**c) Passband group-delay tolerance**

NOTE – The lowest indicated values in b) and c) are for 1 kHz (instead of 0 MHz).

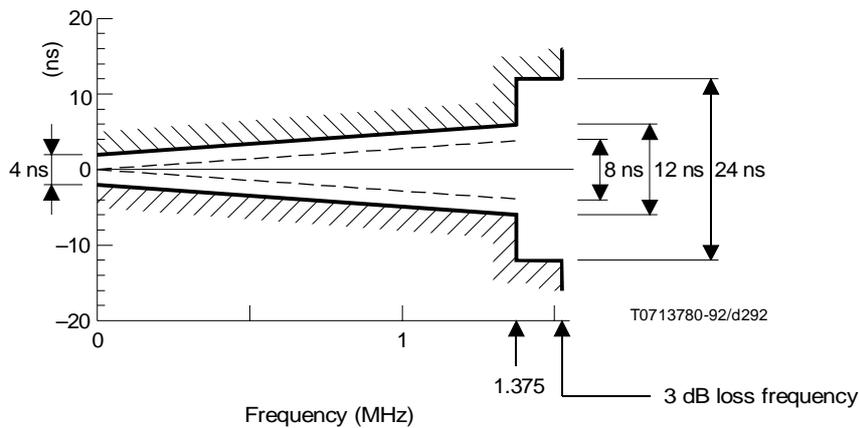
**Table V.4/T.101 – Specification for a colour-difference signal filter used when sampling at 3.375 MHz**



**a) Template for insertion loss/frequency characteristic**



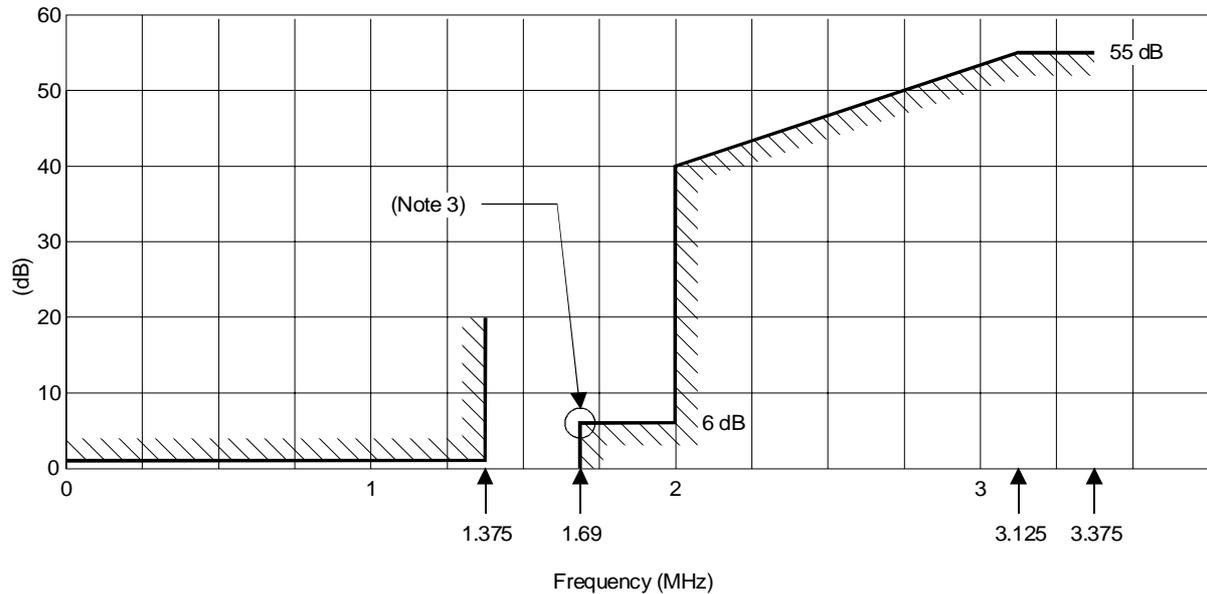
**b) Passband ripple tolerance**



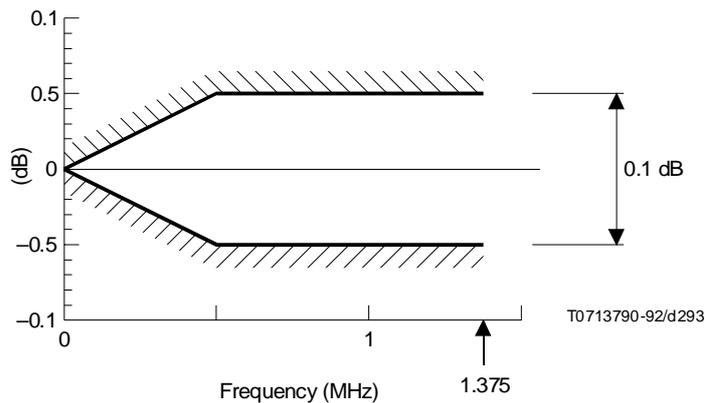
**c) Passband group-delay tolerance**

NOTE – The lowest indicated values in b) and c) are 1 KHz (instead of 0 MHz).

**Table V.5/T.101 – Specification for a digital filter for sampling-rate conversion from 2:2:2 to 2:1:1 colour-difference-signals**



**a) Template for insertion loss frequency characteristic**



**b) Passband ripple tolerance**

NOTES to Table V.3, V.4 and V.5:

- 1 Ripple and group delay are specified relative to their value at 1 kHz. The full lines are practical limits and the dashed lines give suggested limits for the theoretical design.
- 2 In the digital filter, the practical and design limits are the same. The delay distortion is zero, by design.
- 3 In the digital filter (see Table V.5), the amplitude/frequency characteristic (on linear scales) should be skew-symmetrical about the half-amplitude point, which is indicated on the figure.
- 4 In the proposals for the filters used in the encoding and decoding processes, it has been assumed that, in the post-filters which follow digital-to-analogue conversion, correction for the  $(\sin x/x)$  characteristic of the sample-and-hold circuits is provided.

**Appendix VI**  
(Normative)  
(to Recommendation T.101, Annex F)

**Translation modes**

**VI.0 Mode 0 (No translation, full transparency)**

Under this scheme, no translation of data shall be performed.

**VI.1 Mode 1 (No translation except US)**

Under this scheme, no translation of data is performed, except that all US (01/15) characters in the photographic data are represented by two contiguous US characters in the transmitted data stream.

**VI.2 Mode 2 (3-in-4 coding)**

Each group of three bytes in the photographic data is mapped onto four bytes for transmission as shown in Table VI.1. Any remaining group of one or two bytes at the end of a block of data is mapped onto two or three bytes respectively, with undefined bits set to zero.

**Table VI.1/T.101 – 3-in-4 coding scheme**

Photographic Data	3 bytes			3 bytes			1, 2 or 3 bytes	
Transmitted	4 bytes			4 bytes			2, 3 or 4 bytes	
Within each group, the bits are mapped as follows, where $b_{xy}$ denotes bit $y$ of byte $x$ in the Photographic Data. Bit 8 is not taken into account by this scheme, but may be determined by characteristics of the transmission path in use (for example if parity is required).								
<i>a) Three bytes of Photographic Data</i>								
Transmitted	$b_7$	$b_6$	$b_5$	$b_4$	$b_3$	$b_2$	$b_1$	$b_0$
1st byte	x	1	$b_{17}$	$b_{16}$	$b_{27}$	$b_{26}$	$b_{37}$	$b_{36}$
2nd byte	x	1	$b_{15}$	$b_{14}$	$b_{13}$	$b_{12}$	$b_{11}$	$b_{10}$
3rd byte	x	1	$b_{25}$	$b_{24}$	$b_{23}$	$b_{22}$	$b_{21}$	$b_{20}$
4th byte	x	1	$b_{35}$	$b_{34}$	$b_{33}$	$b_{32}$	$b_{31}$	$b_{30}$
<i>b) Two bytes of photographic data at the end of a sequence</i>								
Transmitted	$b_7$	$b_6$	$b_5$	$b_4$	$b_3$	$b_2$	$b_1$	$b_0$
1st byte	x	1	$b_{17}$	$b_{16}$	$b_{27}$	$b_{26}$	0	0
2nd byte	x	1	$b_{15}$	$b_{14}$	$b_{13}$	$b_{12}$	$b_{11}$	$b_{10}$
3rd byte	x	1	$b_{25}$	$b_{24}$	$b_{23}$	$b_{22}$	$b_{21}$	$b_{20}$
<i>c) One byte of photographic data at the end of a sequence</i>								
Transmitted	$b_7$	$b_6$	$b_5$	$b_4$	$b_3$	$b_2$	$b_1$	$b_0$
1st byte	x	1	$b_{17}$	$b_{16}$	0	0	0	0
2nd byte	x	1	$b_{15}$	$b_{14}$	$b_{13}$	$b_{12}$	$b_{11}$	$b_{10}$

**VI.3 Mode 3 (Shift scheme – 8 bits)**

This mode is not used for photographic data.

#### VI.4 Mode 4 (Shift scheme - 7 bits)

In this scheme, bytes of photographic data are each mapped onto one or two bytes of transmitted data, as shown in Table VI.2. In mode 4, the most significant bit of each byte is not taken into account.

NOTE – Most of the conversions are optional. A decoder should be prepared to accept any mixture of converted or unconverted data in these cases.

**Table VI.2/T.101 – Code conversion for 7-bits shift scheme (mode 4)**

Photographic data	Sender's Conversion	O/M	Transmitted data
0/0-1/14	7/14, x+80 (X+50 hex)	O	7/14, 5/0-6/14
01/15	7/14, 6/15	M	7/14, 6/15
2/0	7/13	O	7/13
2/1-7/10	None	–	2/1-7/10
7/11-7/15	7/11, x–88 (X–58 hex)	M	7/11, 2/3-2/7
8/0-13/0	7/11, x–88 (X–58 hex)	M	7/11, 2/8-7/8
13/1-15/15	7/14, x–88 (X+58 hex)	M	7/14, 2/1-4/15
– Irrelevant O Optional M Mandatory NOTE – In mode 4 the transmitted bytes may have the most significant bit set.			

#### VI.5 Mode 5 (no translation except specific characters)

In this scheme, no translation of data is performed, except that all of the codes in the photographic data stream identified in Table VI.3 are replaced by the code DLE (1/10) followed by the replacement code.

**Table VI.3/T.101 – Code conversion for mode 5**

Code	Replacement code	Transmitted
0/1	4/1	1/10 4/1
0/2	4/2	1/10 4/2
0/3	4/3	1/10 4/3
0/4	4/4	1/10 4/4
0/5	4/5	1/10 4/5
0/6	4/6	1/10 4/6
0/7	4/7	1/10 4/7
1/0	5/0	1/10 5/0
1/1	5/1	1/10 5/1
1/2	5/2	1/10 5/2
1/3	5/3	1/10 5/3
1/4	5/4	1/10 5/4
1/5	5/5	1/10 5/5
1/6	5/6	1/10 5/6
1/7	5/7	1/10 5/7
1/8	5/8	1/10 5/8
1/9	5/9	1/10 5/9
1/10	5/10	1/10 5/10
1/11	5/11	1/10 5/11
1/15	5/15	1/10 5/15

NOTE – This mode 5 is intended to be used in an 8-bit environment but is not usable with transmission procedures which use as control codes two bytes sequences starting with DLE.

**Appendix VII**  
(Informative)  
(to Recommendation T.101, Annex F)

**Huffman tables for the “special spectral selection”**

**VII.1 Introduction**

Huffman tables are defined as it is done in the interchange format defined in JPEG, in terms of:

- a 16-bytes list giving the number of codes of each codelength from 1 to 16 (list of codelength);
- a list of 8-bits values which are assigned to each code. The values are placed in the list in order of increasing codelength (list of values).

These two lists are followed by a table specifying the Huffman codes. Annex C of the JPEG specification explains how to build a Huffman code table from the two lists described above.

The Huffman codes are given as 16-bits hexadecimal values, with the root of the code placed toward the MSB. Only the number of bits defined in “codelength” have to be considered.

For example, considering the luminance DC differences Huffman codes:

**Table VII.1/T.101**

Category	Codelength	Codeword
0	3	4000 0100 0000 0000 0000 Codeword = 010
11	9	FF00 1111 1111 0000 0000 Codeword = 1111 1111 0

**VII.2 Spectral bands**

- Luminance: (0,0) (1,5) (6,14) (15,63)
- Chrominance: (0,0) (1,5) (6,63)

**VII.3 Luminance DC differences**

**VII.3.1 List of codelengths**

0 1 5 1 1 1 1 1 1 0 0 0 0 0 0 0

**VII.3.2 List of values**

3 0 1 2 4 5 6 7 8 9 10 11

**Table VII.2/T.101**

Category	Codelength	Codeword
0	3	4000
1	3	6000
2	3	8000
3	2	0
4	3	A000
5	3	C000
6	4	E000
7	5	F000
8	6	F800
9	7	FC00
10	8	FE00
11	9	FF00

**VII.4 Chrominance DC differences**

**VII.4.1 List of codelengths**

0 2 3 1 1 1 1 1 1 1 0 0 0 0 0 0

**VII.4.2 List of values**

0 1 2 3 4 5 6 7 8 9 10 11

**Table VII.3/T.101**

Category	Codelength	Codeword
0	2	0
1	2	4000
2	3	8000
3	3	A000
4	3	C000
5	4	E000
6	5	F000
7	6	F800
8	7	FC00
9	8	FE00
10	9	FF00
11	10	FF80

**VII.5 Luminance AC coefficients**

**VII.5.1 List of codelengths**

0 2 1 2 3 4 7 4 6 6 6 3 1 1 1 129

**VII.5.2 List of values**

1	2	3	4	17	5	18	33	6	49	64	80	0	16	19	32
48	65	81	7	34	97	113	20	50	96	129	145	161	8	35	66
112	177	193	21	82	98	114	128	209	36	51	130	144	225	240	9
10	22	23	24	25	26	37	38	39	40	41	42	52	53	54	55
56	57	58	67	68	69	70	71	72	73	74	83	84	85	86	87
88	89	90	100	101	102	103	104	105	106	115	116	117	118	119	120
121	122	131	132	133	134	135	136	137	138	146	147	148	149	150	151
152	153	154	160	162	163	164	165	166	167	168	169	170	176	178	179
180	181	182	183	184	185	186	192	194	195	196	197	198	199	200	201
202	208	210	211	212	213	214	215	216	217	218	224	226	227	228	229
230	231	232	233	234	241	242	243	244	245	246	247	248	249	250	

**Table VII.4/T.101**

Values	Run/Size	Codelength	Codeword
0	0/0	7	E800
1	0/1	2	0
2	0/2	2	4000
3	0/3	3	8000
4	0/4	4	A000
5	0/5	5	C000
6	0/6	6	D800
7	0/7	8	F600
8	0/8	10	FD00
9	0/9	16	FF7E
10	0/A	16	FF7F
16	1/0	7	EA00
17	1/1	4	B000
18	1/2	5	C800
19	1/3	7	EC00
20	1/4	9	FA00
21	1/5	11	FE80
22	1/6	16	FF80
23	1/7	16	FF81
24	1/8	16	FF82
25	1/9	16	FF83
26	1/A	16	FF84
32	2/0	7	EE00
33	2/1	5	D000
34	2/2	8	F700
35	2/3	10	FD40
36	2/4	12	FF40
37	2/5	16	FF85
38	2/6	16	FF86
39	2/7	16	FF87
40	2/8	16	FF88
41	2/9	16	FF89
42	2/A	16	FF8A
48	3/0	7	F000
49	3/1	6	DC00
50	3/2	9	FA80
51	3/3	12	FF50
52	3/4	16	FF8B

**Table VII.4/T.101** (continued)

Values	Run/Size	Codelength	Codeword
53	3/5	16	FF8C
54	3/6	16	FF8D
55	3/7	16	FF8E
56	3/8	16	FF8F
57	3/9	16	FF90
58	3/A	16	FF91
64	4/0	6	E000
65	4/1	7	F200
66	4/2	10	FD80
67	4/3	16	FF92
68	4/4	16	FF93
69	4/5	16	FF94
70	4/6	16	FF95
71	4/7	16	FF96
72	4/8	16	FF97
73	4/9	16	FF98
74	4/A	16	FF99
80	5/0	6	E400
81	5/1	7	F400
82	5/2	11	FEA0
83	5/3	16	FF9A
84	5/4	16	FF9B
85	5/5	16	FF9C
86	5/6	16	FF9D
87	5/7	16	FF9E
88	5/8	16	FF9F
89	5/9	16	FFA0
90	5/A	16	FFA1
96	6/0	9	FB00
97	6/1	8	F800
98	6/2	11	FEC0
99	6/3	16	FFA2
100	6/4	16	FFA3
101	6/5	16	FFA4
102	6/6	16	FFA5
103	6/7	16	FFA6
104	6/8	16	FFA7
105	6/9	16	FFA8
106	6/A	16	FFA9
112	7/0	10	FDC0
113	7/1	8	F900
114	7/2	11	FEE0
115	7/3	16	FFAA
116	7/4	16	FFAB
117	7/5	16	FFAC
118	7/6	16	FFAD
119	7/7	16	FFAE

**Table VII.4/T.101** (continued)

Values	Run/Size	Codelength	Codeword
120	7/8	16	FFAF
121	7/9	16	FFB0
122	7/A	16	FFB1
128	8/0	11	FF00
129	8/1	9	FB80
130	8/2	12	FF60
131	8/3	16	FFB2
132	8/4	16	FFB3
133	8/5	16	FFB4
134	8/6	16	FFB5
135	8/7	16	FFB6
136	8/8	16	FFB7
137	8/9	16	FFB8
138	8/A	16	FFB9
144	9/0	13	FF70
145	9/1	9	FC00
146	9/2	16	FFBA
147	9/3	16	FFBB
148	9/4	16	FFBC
149	9/5	16	FFBD
150	9/6	16	FFBE
151	9/7	16	FFBF
152	9/8	16	FFC0
153	9/9	16	FFC1
154	9/A	16	FFC2
160	A/0	16	FFC3
161	A/1	9	FC80
162	A/2	16	FFC4
163	A/3	16	FFC5
164	A/4	16	FFC6
165	A/5	16	FFC7
166	A/6	16	FFC8
167	A/7	16	FFC9
168	A/8	16	FFCA
169	A/9	16	FFCB
170	A/A	16	FFCC
176	B/0	16	FFCD
177	B/1	10	FE00
178	B/2	16	FFCE
179	B/3	16	FFCF
180	B/4	16	FFD0
181	B/5	16	FFD1
182	B/6	16	FFD2
183	B/7	16	FFD3
184	B/8	16	FFD4
185	B/9	16	FFD5

**Table VII.4/T.101 (end)**

Values	Run/Size	Codelength	Codeword
186	B/A	16	FFD6
192	C/0	16	FFD7
193	C/1	10	FE40
194	C/2	16	FFD8
195	C/3	16	FFD9
196	C/4	16	FFDA
197	C/5	16	FFDB
198	C/6	16	FFDC
199	C/7	16	FFDD
200	C/8	16	FFDE
201	C/9	16	FFDF
202	C/A	16	FFE0
208	D/0	16	FFE1
209	D/1	11	FF20
210	D/2	16	FFE2
211	D/3	16	FFE3
212	D/4	16	FFE4
213	D/5	16	FFE5
214	D/6	16	FFE6
215	D/7	16	FFE7
216	D/8	16	FFE8
217	D/9	16	FFE9
218	D/A	16	FFEA
224	E/0	16	FFEB
225	E/1	14	FF78
226	E/2	16	FFEC
227	E/3	16	FFED
228	E/4	16	FFEE
229	E/5	16	FFEF
230	E/6	16	FFF0
231	E/7	16	FFF1
232	E/8	16	FFF2
233	E/9	16	FFF3
234	E/A	16	FFF4
240	F/0	15	FF7C
241	F/1	16	FFF5
242	F/2	16	FFF6
243	F/3	16	FFF7
244	F/4	16	FFF8
245	F/5	16	FFF9
246	F/6	16	FFFA
247	F/7	16	FFFB
248	F/8	16	FFFC
249	F/9	16	FFFD
250	F/A	16	FFFE

## VII.6 Chrominance AC coefficients

### VII.6.1 List of codelengths

0 1 2 4 1 5 11 7 8 6 4 8 1 0 1 117

### VII.6.2 List of values

1	2	3	4	5	17	80	6	7	18	32	33	49	0	16	19
48	50	64	65	81	96	97	113	8	20	34	112	129	145	177	9
35	51	66	128	161	193	240	21	82	98	114	209	10	36	160	225
22	52	67	146	176	178	210	226	37	83	23	24	25	26	38	39
40	41	42	53	54	55	56	57	58	68	69	70	71	72	73	74
84	85	86	87	88	89	90	99	100	101	102	103	104	105	106	115
116	117	118	119	120	121	122	130	131	132	133	134	135	136	137	138
147	148	149	150	151	152	153	154	162	163	164	165	166	167	168	169
170	179	180	181	182	183	184	185	186	192	194	195	196	197	198	199
200	201	202	208	211	212	213	214	215	216	217	218	224	227	228	229
230	231	232	233	234	241	242	243	244	245	246	247	248	249	250	

**Table VII.5/T.101**

Values	Run/Size	Codelength	Codeword
0	0/0	7	DC00
1	0/1	2	0
2	0/2	3	4000
3	0/3	3	6000
4	0/4	4	8000
5	0/5	4	9000
6	0/6	5	C000
7	0/7	6	C800
8	0/8	8	F200
9	0/9	9	F900
10	0/A	11	FE80
16	1/0	7	DE00
17	1/1	4	A000
18	1/2	6	CC00
19	1/3	7	E000
20	1/4	8	F300
21	1/5	10	FD00
22	1/6	12	FF00
23	1/7	16	FF8A
24	1/8	16	FF8B
25	1/9	16	FF8C
26	1/A	16	FF8D
32	2/0	6	D000
33	2/1	6	D400
34	2/2	8	F400
35	2/3	9	F980
36	2/4	11	FEA0
37	2/5	13	FF80
38	2/6	16	FF8E
39	2/7	16	FF8F
40	2/8	16	FF90

**Table VII.5/T.101** (continued)

Values	Run/Size	Codelength	Codeword
41	2/9	16	FF91
42	2/A	16	FF92
48	3/0	7	E200
49	3/1	6	D800
50	3/2	7	E400
51	3/3	9	FA00
52	3/4	12	FF10
53	3/5	16	FF93
54	3/6	16	FF94
55	3/7	16	FF95
56	3/8	16	FF96
57	3/9	16	FF97
58	3/A	16	FF98
64	4/0	7	E600
65	4/1	7	E800
66	4/2	9	FA80
67	4/3	12	FF20
68	4/4	16	FF99
69	4/5	16	FF9A
70	4/6	16	FF9B
71	4/7	16	FF9C
72	4/8	16	FF9D
73	4/9	16	FF9E
74	4/A	16	FF9F
80	5/0	4	B000
81	5/1	7	EA00
82	5/2	10	FD40
83	5/3	15	FF88
84	5/4	16	FFA0
85	5/5	16	FFA1
86	5/6	16	FFA2
87	5/7	16	FFA3
88	5/8	16	FFA4
89	5/9	16	FFA5
90	5/A	16	FFA6
96	6/0	7	EC00
97	6/1	7	EE00
98	6/2	10	FD80
99	6/3	16	FFA7
100	6/4	16	FFA8
101	6/5	16	FFA9
102	6/6	16	FFAA
103	6/7	16	FFAB
104	6/8	16	FFAC
105	6/9	16	FFAD
106	6/A	16	FFAE

**Table VII.5/T.101** (continued)

Values	Run/Size	Codelength	Codeword
112	7/0	8	F500
113	7/1	7	F000
114	7/2	10	FDC0
115	7/3	16	FFAF
116	7/4	16	FFB0
117	7/5	16	FFB1
118	7/6	16	FFB2
119	7/7	16	FFB3
120	7/8	16	FFB4
121	7/9	16	FFB5
122	7/A	16	FFB6
128	8/0	9	FB00
129	8/1	8	F600
130	8/2	16	FFB7
131	8/3	16	FFB8
132	8/4	16	FFB9
133	8/5	16	FFBA
134	8/6	16	FFBB
135	8/7	16	FFBC
136	8/8	16	FFBD
137	8/9	16	FFBE
138	8/A	16	FFBF
144	9/0	10	FE00
145	9/1	8	F700
146	9/2	12	FF30
147	9/3	16	FFC0
148	9/4	16	FFC1
149	9/5	16	FFC2
150	9/6	16	FFC3
151	9/7	16	FFC4
152	9/8	16	FFC5
153	9/9	16	FFC6
154	9/A	16	FFC7
160	A/0	11	FEC0
161	A/1	9	FB80
162	A/2	16	FFC8
163	A/3	16	FFC9
164	A/4	16	FFCA
165	A/5	16	FFCB
166	A/6	16	FFCC
167	A/7	16	FFCD
168	A/8	16	FFCE
169	A/9	16	FFCF
170	A/A	16	FFD0
176	B/0	12	FF40
177	B/1	8	F800

**Table VII.5/T.101** (continued)

Values	Run/Size	Codelength	Codeword
178	B/2	12	FF50
179	B/3	16	FFD1
180	B/4	16	FFD2
181	B/5	16	FFD3
182	B/6	16	FFD4
183	B/7	16	FFD5
184	B/8	16	FFD6
185	B/9	16	FFD7
186	B/A	16	FFD8
192	C/0	16	FFD9
193	C/1	9	FC00
194	C/2	16	FFDA
195	C/3	16	FFDB
196	C/4	16	FFDC
197	C/5	16	FFDD
198	C/6	16	FFDE
199	C/7	16	FFDF
200	C/8	16	FFE0
201	C/9	16	FFE1
202	C/A	16	FFE2
208	D/0	16	FFE3
209	D/1	10	FE40
210	D/2	12	FF60
211	D/3	16	FFE4
212	D/4	16	FFE5
213	D/5	16	FFE6
214	D/6	16	FFE7
215	D/7	16	FFE8
216	D/8	16	FFE9
217	D/9	16	FFEA
218	D/A	16	FFEB
224	E/0	16	FFEC
225	E/1	11	FEE0
226	E/2	12	FF70
227	E/3	16	FFED
228	E/4	16	FFEE
229	E/5	16	FFEF
230	E/6	16	FFF0
231	E/7	16	FFF1
232	E/8	16	FFF2
233	E/9	16	FFF3
234	E/A	16	FFF4
240	F/0	9	FC80
241	F/1	16	FFF5
242	F/2	16	FFF6
243	F/3	16	FFF7

**Table VII.5/T.101 (end)**

Values	Run/Size	Codelength	Codeword
244	F/4	16	FFF8
245	F/5	16	FFF9
246	F/6	16	FFFA
247	F/7	16	FFFB
248	F/8	16	FFFC
249	F/9	16	FFFD
250	F/A	16	FFFE

## **Appendix VIII**

(Informative)

(to Recommendation T.101, Annex F)

### **Examples of local presentation facilities with T4 and T6 encoded data**

#### **VIII.1 Local presentation facilities**

When ITU-T Recommendation T.4 or CCITT Recommendation T.6 coding algorithms are used to present encoded facsimile photographic images, with Annex F, Photographic Data Syntax, it is possible to transmit encoded facsimile photographic images larger than the size of the corresponding specified photo-area and possibly larger than the physical DDA (see F.7.3.2).

In this case local facilities can be offered to present the whole facsimile data content to the user, **after an initial presentation in conformance with F.7.3.2, either:**

- by the use of local scrolling facilities within the photo-area;
- by the use of one or several of the following local presentation modes in the photo area or in the full physical DDA; or
- or by a combination of local scrolling facilities and of the following local presentation modes in the photo area or in the full physical DDA.

#### **VIII.2 Local presentation modes**

The following local presentation modes define how a facsimile encoded picture contained in the Source Defined Display Area (SDDA) can be presented on the user output display after an initial presentation in conformance with F.7.3.2.

The local decision to present an encoded facsimile photographic picture in one of these modes, after an initial presentation in conformance with F.7.3.2, and the choice between these modes if several are provided, should be offered by the output display to the user.

In all cases, the visual aspect of the pixel density ratio shall be maintained between the source and the output display.

The presentation modes are the following:

- full view ;
- full width ;
- full height ;
- original size ;
- terminal dependant.

### VIII.2.1 Full view mode

In full view mode, the entire facsimile picture of the SDDA shall be made visible on the output display either in the photo-area or in the full physical DDA. This mode can be used to display an overview of the whole facsimile picture.

NOTE – When the source aspect ratio, and/or the source pixel density ratio, and/or the number of horizontal and/or vertical pixels of the SDDA are very different from the ones of the output display, there is no guarantee that the entire information will be still fully readable.

#### Example:

When the source is a G3 facsimile scanner and the output display is a TV screen in the full physical DDA mode, the source aspect ratio, the source pixel density ratio, the number of horizontal and vertical pixels are very different between the source and the output display.

An A4 page SDDA, scanned by a G3 facsimile apparatus in standard mode, represents 1728 pixels horizontally and 1143 pixels vertically with a source aspect ratio of  $215\text{ mm}/297\text{ mm} = 0.724$ . The source pixel density ratio is  $8.037\text{ ppm}/3.85\text{ ppm} = 2.088$ . The visual aspects of the source aspect ratio and source pixel density ratio shall be maintained when displaying the image on the output device. Typically, for a TV set display in full DDA, the output aspect ratio is 1.333 (4/3), with 640 pixels horizontally and 480 vertically the output pixel density ratio is 1.

It is then up to the terminal to make the conversion so that the SDDA fits entirely within the full physical DDA. Along the vertical line, the 1143 source pixels shall be mapped on 480 pixels, each pixel representing  $1143/480 = 2.381$  source pixels. Along the horizontal axis, the whole width shall be visible with the source aspect ratio of the image being maintained. Therefore a maximum of  $480 \times 215/297 = 347$  pixels shall be used. One may also say that in order to maintain the pixel aspect ratio from the source to the output device, the 1143 ( $297 \times 3.85$ ) vertical source pixels then represents  $1143 \times 2.088 = 2386$  virtual pixels whose aspect ratio are those of the output. Those 2386 virtual pixels are mapped onto the 480 vertical pixels, each output pixel representing  $2386/480 = 4.971$  virtual pixels. Along the horizontal axis, a maximum of  $1728/4.971$  pixels shall be used that is to say: 347 pixels, each output pixel representing 4.971 source pixels.

### VIII.2.2 Full width mode

In full width mode, the facsimile picture shall be presented on the output display, either in the photo-area or in the full physical DDA, in such a way that the entire width of the facsimile picture is exactly mapped with the entire width of the photo-area or of the physical DDA.

NOTE – There is no guarantee that the facsimile picture information presented on the output display will be still fully readable. It may also turn out that this presentation mode gives exactly the same result as the “full view” presentation mode.

#### Example:

With the figures of the example in VIII.2.1, the 640 horizontal pixels of the full physical DDA shall represent the original 1728 horizontal pixels, each output pixel representing 2.70 source pixels. On the vertical axis, the 1143 source pixels represent 2386 virtual pixels to be mapped onto  $2386/2.7 = 883$  output pixels. The 480 vertical pixels of the DDA will then represent  $480 \times 2.70 = 1296$  virtual pixels, i.e.  $(1143/883) \times 480 = 621$  source pixels.

### VIII.2.3 Full height mode

In full height mode, the facsimile picture shall be presented on the output display, either in the photo-area or in the full physical DDA, in such a way that the entire height of the facsimile picture is exactly mapped with the entire height of the photo-area or of the physical DDA.

NOTE – There is no guarantee that the facsimile picture information presented on the output display will be still fully readable. It may also turn out that this presentation mode gives exactly the same result as the “full view” presentation mode.

#### Example:

With the figures of the example in VIII.2.1, the output image is similar with the one of the example in VIII.2.1 above.

#### **VIII.2.4 Original size mode**

In original size mode, the output display, either the photo-area or the full physical DDA shall be mapped with the maximum of possible facsimile picture information without any enlargement or reduction (one horizontal source pixel for one horizontal output pixel). However, the visual aspect of the pixel density ratio shall be maintained.

#### **Example:**

With the figures of the example in VIII.2.1, the 640 horizontal pixels of the full physical DDA shall represent the original 640 left horizontal source pixels. On the vertical axis, the 480 pixels of the output display area will then represent  $480/2.088 = 230$  vertical pixels of the source information.

#### **VIII.2.5 Terminal dependant mode**

In terminal dependent mode, it is up to the terminal to present a readable facsimile picture, enlarged or reduced, on the output display either in the photo-area or in the full physical DDA. The definition and use of local facilities to enlarge, to reduce, to scroll the source facsimile picture information is always possible but is beyond the scope of this photographic data syntax.