ANNEX A

(To Recommendation T.101)

Interworking data syntax (IDS) described in ASN.1 (Recommendation X.208)

Preamble

For Videotex interchange:

a) If two countries implement the same data syntax, then interworking can use the same data syntax (DS I, or DS II, or DS III).

b) If two countries implement two different data syntaxes, then interworking can use either:

   i) the interworking data syntax (IDS) as defined herein, or

   ii) any one of the three data syntaxes and convert directly between DS I/DS II/DS III. The data syntaxes may be identified by the ESC 2/5 F mechanism described in § 4.4.2 of the main body of Recommendation T.101.

If the IDS is used, then the Administration in which country the data base is located will be responsible to convert into the IDS and the Administration in which country the user terminal is located will be responsible to convert from the IDS.

If the direct conversion method is used instead of using the IDS, the IDS would serve as a technical guide in designing the conversion process.

The IDS is not intended to be used in terminal to host communications.

A.1 *Videotex page*

A Videotex page in the interworking data syntax (IDS) is a sequence of presentation commands expressed in a manner independent of any of the terminal data syntaxes. This formulation of the presentation information which composes a Videotex page is intended to aid interworking between basically different terminal data syntaxes. It does this by isolating the unique and common elements between each of the data syntaxes. The interworking data syntax is not meant to be used as a terminal data syntax in its own right. One encoding of the interworking data syntax is that defined in Recommendation X.209. Other types of encoding are for further study.

```
Videotex—Page          ::= SEQUENCE OF Presentation—Commands

Presentation—Commands ::= SEQUENCE { State—Vector, Function—&—Parameters }
```

A.2 *State vector*

A state victor is defined along with each presentation command to establish the relationship of that presentation command to each other presentation command. Although the information explicitly contained in the state vector is also implicitly contained within each presentation command, it would require the conversion apparatus to fully understand each of the three terminal oriented data syntaxes to uncover this information. Therefore a state vector is included with each presentation command in which a global state is affected or in which a boundary value is encountered, so that the conversion process might operate on a general level.

```
State—Vector           ::= CHOICE { [1] Vector—Definition
                                    [2] Reset—State—Vector,
                                    [3] NULL }
```

A.2.1 *Vector definition*

```
Vector—Definition          ::= SEQUENCE { Global—State—Affected—Indicator,
                                  Terminal—Model—Precedence,
                                  Boundary—Condition—Definition }
```

—— Only that information which changes between state vectors need be communicated. If there is no change in a particular component of the state vector, then that component need not be communicated. This means that the state vector is not communicated often and does not introduce significant overhead.

A.2.1.1 *Global state affected indicator*

The global state affected indicator carries information relating to the global states of the presentation data syntax. Global state variables are variables representing those states of the presentation data syntax which are established by presentation commands and which carry on to affect the results of subsequent presentation commands. By declaring the global state variables explicitly, it is not necessary for the conversion process to undarstand the interrelationship between presentation commands. This means that the conversion process does not have to simulate a terminal of the source data syntax in order to handle the conversion of its elements.

The global state affected indicator does not carry information about the value to which a global state has been set. That information is carried within the `Function and Parameter` section of the IDS. The indicator merely identifies which states have changed. This is of great importance in situations where it is necessary for the conversion process to sort the presentation commands to account for differences in the terminal model used in the source and destination of the interchange. If the order of presentation commands is altered, the conversion process must establish the appropriate global variables before each command in the altered sequence. By referring to the global state affected indicator, the conversion process can determine which global states must be re—established. For example, if a sorting of presentation commands is necessary to convert from a multi—plane to a single plane terminal mode, and colour control commands have been used, then the global state affected indicator will indicate that the appropriate colour state must be established ahead of each portion of the sorted data.

In some of the terminal data syntaxes attributes have global effects while in other data syntaxes the effects are localized according to the particular display primitive type. For example, in Data Syntax III a colour command remains in effect for all primitives, until the next colour command, whereas in Data Syntax II there are various colour states which apply independently to different primitives, such as LINE colour, FILLED AREA colour, etc. The global state indicator carries a reference to a number of independent `attribute state vectors` which define the attribute context. For those data syntaxes which make use of only global parameters, only one `attribute state vector` need be referenced. For other data syntaxes which make use of multiple localized attributes, several `attribute state vectors` may be referenced.

```
Global—State—Affected—Indicator ::= SEQUENCE {
attribute—state—vector—reference            INTEGER,
attribute—affected—indicators           SEQUENCE OF {
                                            INTEGER {
                                        current—text—position                   (1),
                                        current—foreground—colour       (2),
                                        current—auxiliary—colour        (3),
                                        lining—state                    (4),
                                        flash—blink—state               (5),
                                        basic—char—size—state                   (6),
                                        conceal—state                   (7),
                                        char—invert—box—state           (8),
                                        char—marking—state                      (9),
                                        screen—protection—state         (10),
                                        display—control—state                   (11),
                                        device—control—state                    (12),
                                        cursor—control—state                    (13),
                                        geometric—control—1—state       (14),
                                        geometric—control—2—state       (15),
                                        wait—state                      (16),
                                        general—text—state              (17),
                                        p—text—state                            (18),
                                        geometric—text—state                    (19),
                                        DRCS—definition—state           (20),
                                        macro—definition—state          (21),
                                        texture—pattern—state                   (22),
                                        music—part—memory—state         (23),
                                        animation—configuration—state           (24),
                                        workstation—configuration—state (25)
                                    } } }
```

—— The Global State Affected Indicator consists of a set of indicator flags which identify particular global states or in some cases categories of global states which may be altered by presentation and control commands. Some states, such as all the forms of Flashing and Blink processes, are grouped together for simplicity.

A.2.1.2 *Terminal model*

The terminal model differs significantly between the various terminal data syntaxes. For the presentation of static images this manifests itself in the manner by which presentation commands overlay each other. A picture developed for a multi—plane terminal model can be represented on a terminal using a single—plane terminal model or a multi—plane terminal model with a different order of precedence for the planes, by sorting the presentation commands so that they build up an equivalent picture. The sorting operation is necessary since otherwise the buildup order might conflict with the precedence order in the new environment. The terminal model precedence indicator is simply a numeric indicator of the overlay precedence for presentation commands intended by the source terminal data syntax. The conversion process is independent of the terminal model or a particular data syntax and simply sorts presentation commands based on this indicator. Note that certain commands such as resets which have an effect in more than one plane of a terminal model might have to be repeated in different parts of the presentation sequence after the sort. The terminal model precedence indicator consists of a sequence of numbers to indicate the effect of a command across the terminal model.

Terminal—Model—Precedence ::= INTEGER

—— The terminal model precedence indicator is a number which indicates the order of precedence of the identified presentation information. The number `1` indicates that the identified information is of highest precedence and should be placed in front of any other information currently displayed. The number `2` indicates that the identified information is of the second level of precedence and should be placed behind any level `1` information but ahead of any other level information. For example, Data Syntax II Text and Mosaic data is level `1` information, whereas geometric information may be level `1` or level `2`. For Data Syntax III, all of the information is at level `1` since the precedence order by which it is displayed is determined only by the order in which the information is communicated. For Data Syntax I, the order is not fixed since certain `planes` of memory may be changed in precedence by the ASSIGN FRAME command.

—— The value `0` has special meaning for the Terminal Model Precedence Indicator. It indicates that the identified information requires special interpretation. Such special information includes partial reset commands which affect more than one layer of the terminal model, as well as commands having a time dependent effect, specifically WAIT, the BEL character, and REVEAL.

A.2.1.3 *Boundary conditions*

Boundary condition variables represent the limits within which the particular presentation command has been defined. Each presentation command takes on its normal interpretation only within a certain range of values. For example, the number of characters which may be displayed on the screen varies between each of the source terminal data syntaxes, and therefore the operation of presenting a single character cannot be considered to be the same in each terminal data syntax. To factor out the commonality, the boundary condition of encountering the edge of the display area is identified separately from the presentation of a character. This aids conversion since it means that the boundary conditions applying to each presentation command are given explicitly. The conversion process is therefore independent of the internal boundary conditions within each of the source terminal data syntaxes.

Boundary—Condition—Definition ::= SET { [1] Screen—Dimensions,

[2] Colour—Map—Limit,

[3] Presentation—Sub—Area,

[4] Char—Mode—Constraints,

[5] Coordinate—Limit—Polygon,

[6] Coordinate—Limit—Spline,

[7] Presentation—Resolution,

[8] Macro—Seg—Memory—Limit,

[9] DRCS—Memory—Limit,

[10] Direct—Colours—Limit }

A.2.1.3.1 *Screen dimensions*

Screen—Dimensions ::= SEQUENCE { INTEGER, INTEGER }

—— Screen—Dimensions indicates the aspect ratio of the display screen expressed in terms of a fraction of the Y and a fraction of the X unit dimensions, where the INTEGER number represents a binary fraction with an implied binary point before the most significant bit. Note that a dimension of (1,1) implies no geometric constraint. A character mode service could use (1,1) to imply no constraint.

A.2.1.3.2 *Colour map limit*

Colour—Map—Limit ::= INTEGER

—— The colour map limit indicates the maximum number of colours which may be stored in a single colour map, or the combined total for multiple colour maps, and represents the maximum number of colour states which may be encountered in a particular presentation page. In the case where no colour map is used, the integer specifies the number of fixed colours.

A.2.1.3.3 *Presentation sub—area*

Presentation—Sub—Area ::= SEQUENCE { Abs—Coord, Rel—Coord, INTEGER, INTEGER }

—— The two coordinates give the boundary dimensions of a sub—area of the display screen both in terms of the dimensions of the sub—area and the number of characters per row and the number of columns. The absolute coordinate specifies the origin of the sub—area, the relative coordinate the size of the sub—area and the INTEGER coordinates the limit on characters per row and rows respectively.

A.2.1.3.4 *Char mode constraints*

Char—Mode—Constraints ::= SEQUENCE { INTEGER, INTEGER }

—— The two parameters give the limit to the number of characters per row and the number of rows of text which may be presented on the display screen; that is, the the boundaries at which character (or word) wrap and scroll will occur.

A.2.1.3.5 *Coordinate limit polygon*

Coordinate—Limit—Polygon ::= INTEGER

—— The polygon coordinate limit specifies the maximum number of coordinates which may be specified for a filled polygon.

A.2.1.3.6 *Coordinate limit spline*

Coordinate—Limit—Spline ::= INTEGER

—— The spline coordinate limit specifies the maximum number of coordinates which may be specified.

A.2.1.3.7 *Presentation resolution*

Presentation—Resolution ::= SEQUENCE { INTEGER, INTEGER }

—— The presentation resolution specifies the nominal resolution of the display screen which was used by the information source.

A.2.1.3.8 *Macro seg memory limit*

Macro—Seg—Memory—Limit ::= INTEGER

—— The macro memory limit specifies the upper bound on the amount of memory which is available for the storage of Macros or Segments. The INTEGER parameter represents available memory expressed in bytes.

A.2.1.3.9 *DRCS memory limit*

DRCS—Memory—Limit ::= INTEGER

—— The DRCS memory limit specifies the upper bound on the amount of memory which is available for the storage of DRCS. The INTEGER parameter represents available memory expressed in bytes.

A.2.1.4 *Data syntax identifier (SID)*

SID ::= IMPLICIT INTEGER { data—syntax— I (1),

data—syntax— II (2),
data—syntax—III (3) }

—— SID is an identifier which is referenced in a number of primitive commands and which identifies the source data syntax of the command.

A.2.2 *Reset state vector*

Reset—State—Vector ::= SEQUENCE { SID, Vector—Definition }

—— The Reset State Vector command is used to establish the initial state for the Interworking Data Syntax. The default state may be selected from the table corresponding to the source terminal data syntax (or profile) given in Appendix II. Alternate parameters may be specified by use of explicit state vector and function and parameter definitions.

A.2.3 *NULL*

NULL implies that the state vector is unchanged from the previous presentation command.

A.3 *Functions and parameters*

The functions and parameters which make up the presentation commands are grouped into categories which depend upon their commonality between the various terminal data syntaxes. Those functions which are compatible, such as the basic repertoire of alphanumeric characters defined in Recommendation T.51, define separate groups. Those functions which are unique, such as certain specific special characters, also establish separate groups so that they may be converted or otherwise handled in a special manner. Functions such as DRCS and graphics drawing commands, which differ in fundamental ways between the various terminal data syntaxes, are organized so that those underlying capabilities which are common may be exploited in the necessary conversion process.

Functions—&—Parameters ::= CHOICE { [0] Alpha—Char—String,
[1] Special—Char—String,
[2] Kana—Char—String,
[3] Kanji—Char—String,
[4] Block—Mosaic—String,
[5] Smooth—Mosaic—String,
[6] Special—Mosaic—String,
[7] Format—Effector—C0—Chars,
[8] Special—Format—C0—Characters,
[9] General—Control—Characters,
[10] Geometric—String,
[11] Animation—Control—String,
[12] Segment—Control—String,
[13] Colour—Control—String,
[14] Text—Control—String,
[15] Photo—Graphic—String—Syntetic—Image,
[16] Photo—Graphic—String—Natural—Image,
[17] MACRO—String,
[18] DRCS—String,
[19] Fill Pattern—Control—String,
[20] Music—String,
[21] Tele—Software—String,
[22] Audio—Data—String,
[23] Greek—Char—String }

The first six categories of functions and the last one are various text or mosaic characters. None of the terminal data syntaxes defined in Recommendation T.101 encompasses all of these characters. There are different unique characters in each of the terminal data syntaxes. However, a large portion of the repertoire is common between the different terminal data syntaxes, although the characters may be coded differently. Since coding is irrelevant here, and the use of particular tables could in fact cause serious confusion, characters extracted from the different character repertoires will be distinguished by the identifier name codes for each character as defined in Recommendation T.51. Since all of the terminal—oriented data syntaxes in Recommendation T.101 do not explicitly make use of these name codes in the body of the Recommendation, the entire character repertoire, together with the name codes for each character are included here as an appendix.

A.3.0 *Alpha char string*

Alpha—Char—String ::= GRAPHICSTRING

—— Characters (LA01 to LZ30, ND01 to ND09 and ND10, SC01 to SC05, SP01 to SP22, SA01 to SA07, NS01 to NS03, NF01 to NF21, SM01 to SM44 and SM47 to SM49, and SD11 to SD43) taken from Repertoire 1 which are the characters from the primary and supplementary character sets of Recommendation T.51 together with the SPACE character (SP01) and DELETE character (SM34).

—— The coding of characters within an Alpha Character String will be taken from the IRV primary character code table (ISO Registration Number 2 under ISO 2375) and the secondary code table for use with IRV from ISO 6937/2 (ISO Registration Number 90).

—— *Note* — The coding for the character $ "Dollar Sign" (SC02) will be taken from the supplementary character set.

—— *Note* — The coding for the character ## "number sign" (SM01) will be taken from the primary character set.

—— *Note* — The coding for the character "general currency sign" (SC01) will be taken from the primary character set.

A.3.1 *Special char string*

```
Special—Char—String     ::= INTEGER { non—spacing—vector—overbar    (1),
                                      non—spacing—slant             (2),
                                      left—vertical—bar—jointive          (3),
                                      right—vertical—bar—jointive   (4) }
```

—— Non—Spacing—Vector—Overbar is a character (SM50) from Repertoire 2.

—— Non—Spacing—Slant is a character (SM51) from Repertoire 2.

—— Left—Vertical—Bar—Jointive is a character (SM45) from Repertoire 2.

—— Right—Vertical—Bar—Jointive is a character (SM46) from Repertoire 2.

A.3.2 *Kana char string*

Kana—Char—String ::= GRAPHICSTRING

—— Characters (JA01 to JA63) taken from Repertoire 3.

—— The coding of characters within a Kana Character String will be taken from the Kana character code table (ISO Registration Number 56 under ISO 2375).

A.3.3 *Kanji char string*

Kanji—Char—String ::= GRAPHICSTRING

—— Characters (JK01 to JK2980, HK01 to HK83, and JS01 to JS366) from Repertoire 4.

—— The coding of characters within a Kanji Character String will be taken from the two byte Kanji character code table (ISO Registration Number 87 under ISO 2375).

—— *Note* — The characters in this two byte code table which overlap other defined videotex character set are not considered to be part of Repertoire 4, and therefore are communicated as characters from Repertoire 1, Repertoire 3 or Repertoire 8 where appropriate. Specifically this involves the Latin alphanumeric characters (LA01 to LZ30), and non—alphabetic characters (ND01 to ND09 and ND00, SC01 to SC05, SP01, SP02, SP04 to SP15, SP17 to SP22, SA01 to SA07, NS02 to NS03, NF01 to NF05, SM01 to SM14, SM19, SM24 to SM34, SM38, SM43, SM44, SM47, SM48, and SD11 to SD43) from Repertoire 1, the Kana characters (JA01 to JA63) from Repertoire 3, the drawing characters (DG01 to DG04, DG13 to DG24, and DG32) from Repertoire 8, which have an alternate coding within the two byte code, but which are included in other Repertoires.

A.3.4 *Block mosaic string*

Block—Mosaic—String ::= GRAPHICSTRING

—— Block Mosaic characters (MG01 to MG63) taken from Repertoire 7.

—— The coding of characters for the Block Mosaic sub—Repertoire is identical between the three terminal data syntaxes defined in CCITT Recommendation T.101. The set is registered with ISO Number 129 under ISO 2375.

A.3.5 *Smooth mosaic string*

```
Smooth—Mosaic—String ::= CHOICE { [1] Sub—Cell—Aligned—Smooth—Mosaics,
                                   [2] General—Smooth—Mosaics }
```

A.3.5.1 *Sub—call aligned smooth mosaics*

```
Sub—Cell—Aligned—Smooth—Mosaics ::= GRAPHICSTRING
```

—— Smooth Mosaic characters (SG01 to SG56) taken from Repertoire 8.

—— The coding of characters for the Sub Cell Aligned Smooth Mosaic sub—Repertoire is identical between the two terminal data syntaxes defined in Recommendation T.101 which makes available these characters. These are registered as Numbers 71 and 72 under ISO 2375.

A.3.5.2 *General smooth mosaics*

```
General—Smooth—Mosaics ::= GRAPHICSTRING
```

—— Smooth Mosaic characters (MS01 to MS28) taken from Repertoire 8.

—— The coding of characters for the General Smooth Mosaic sub—Repertoire is taken from the terminal data syntax in CCITT Recommendation T.101 which makes available these characters. This code table is registered under ISO 2375 as Registration Number 137.

A.3.6 *Special mosaic string*

```
Special—Mosaic—String ::= CHOICE { [1] Drawing—Characters,
                                    [2] Other—Special—Mosaics }
```

A.3.6.1 *Drawing characters*

```
Drawing—Characters          ::= GRAPHICSTRING
```

—— Drawing characters (DG01 to DG50) taken from Repertoire 10.

A.3.6.2 *Other special mosaics*

```
Other—Special—Mosaics ::= INTEGER { open—left—half—oval          (1),
                                    open—right—half—oval         (2),
                                    filled—left—half—oval        (3),
                                    filled—right—half—oval       (4),
                                    reverse—left—half—oval       (5),
                                    reverse—right—half—oval      (6) }
```

—— Open—Left—Half—Oval is a Special Mosaic characters (MS13) from Repertoire 11.

—— Open—Right—Half—Oval is a Special Mosaic characters (MS14) from Repertoire 11.

—— Filled—Left—Half—Oval is a Special Mosaic characters (MS30) from Repertoire 11.

—— Filled—Right—Half—Oval is a Special Mosaic characters (MS29) from Repertoire 11.

—— Reverse—Left—Half—Oval is a Special Mosaic characters (MS15) from Repertoire 11.

—— Reverse—Right—Half—Oval is a Special Mosaic characters (MS31) from Repertoire 11.

The function and parameter categories 7 and 8 contain basic control characters which are used to control the state of presentation of alphanumeric text and mosaic characters (including DRCS). These control characters can be broken down into two categories, format effector control characters and special format control characters. The format effector control characters have basically the same meaning in each of the three terminal data syntaxes. The only difference is how the functions invoked by these control characters interact with the terminal model and display environment of the various terminal data syntaxes; for example, they may apply to only one plane of display in a multi—plane terminal model or to all planes of display. The coding of the format effector characters is also compatible between the terminal data syntaxes.

The special format control characters in category 8, in general have a special meaning which is not shared by all of the data syntaxes. These functions must be specially converted during interworking, even between data syntaxes which appear to assign the same meaning to a particular control function. This is because the terminal model and display environment of the various terminal data syntaxes are quite different. The Bell character is included in this category because it requires special handling due to the timing of presentation. If a sorting of presentation commands is required in the interworking conversion process to accommodate differences in a terminal model, such as the handling of data intended for a multi—plane terminal on a single plane terminal, then the time of presentation of the Bell character must be altered.

A.3.7 *Format effector C0—char*

Format—Effector—C0—Char ::= GRAPHICSTRING

—— Format Effector C0 characters (APB, APF, APD, APU, CS, APR, APH) taken from CCITT Recommendation T.101 DS I, II, III; (C0 code table positions 0/8 to 0/13 and 1/14 respectively)

—— APB — Active Position Back — analogous to ISO 646 (FE_0 BS)

—— APF — Active Position Forward — (FE_1 HT)

—— APD — Active Position Down — (FE_2 LF)

—— APU — Active Position Up — (FE_3 VT)

—— CS — Clear Screen — (FE_4 FF)

—— APR — Active Position Return — (FE_5 CR)

—— APH — Active Position Home

A.3.8 *Special format—C0—char*

```
Special—Format—C0—Char ::= CHOICE { [1] Bell—Character,
                                     [2] Position—Set,
                                     [3] Cancel—Macro,
                                     [4] Non—Selective—Reset,
                                     [5] Cancel—Row }
```

A.3.8.1 *Bell character*

Bell—Character ::= GRAPHICSTRING

—— Special C0 character (BEL) from Recommendation T.101 DS I, III (C0 set positions 0/7).

—— *Note* — This function provides an audio signal to the user of the terminal device. This function is not available in all of the terminal data syntaxes, and cannot be simulated in a reasonable manner.

A.3.8.2 *Position set*

Position—set ::= SEQUENCE { INTEGER, INTEGER }

—— This function provides the equivalent capability of both the Active Position Set command (APS) from Recommendation T.101 DS I, III and the positioning portion of the Active Position Address command (APA) from DS II.

—— The parameters to establish the new screen active position as a count of “current size” character cells from the “home” upper left position.

A.3.8.3 *Cancel macro*

```
Cancel—Macro            ::= GRAPHICSTRING
```

—— Special C0 character [CAN (Macro)] from Recommendation T.101 DS I, III (C0 set positions 1/8).

A.3.8.4 *Non—selective reset*

```
Non—Selective—Reset ::= SEQUENCE { [1] NSR—Code,
                                   [2] Position—Set OPTIONAL }

NSR—Code            ::= GRAPHICSTRING
```

—— Special C0 character (NSR) from Recommendation T.101 DS I, III (C0 set positions 0/15). The positioning parameter sequence is optional.

A.3.8.5 *Cancel row*

```
Cancel—Row          ::= GRAPHICSTRING
```

—— Special C0 characters [CAN (Row)] from Recommendation T.101 DS II (C0 set positions 1/8).

A.3.9 *General control characters*

The function and parameter category 9 contains general control functions which are used to control the general state of presentation. The meaning of these control characters is very dependent upon the terminal model and display environment of the terminal data syntax in which they are used. Transcoding and conversion is required for each of the functions invoked by these control characters. These control characters have been organized into a number of sub categories which correspond to the area of functionality being addressed.

```
General—Control—Characters ::= CHOICE { [1] Other—Format—Effectors,
                                         [2] Lining—Control,
                                         [3] Character—Size—Control,
                                         [4] Flash—Control,
                                         [5] Conceal—Control,
                                         [6] Invert—Control,
                                         [7] Window/Box—Control,
                                         [8] Marking—Control,
                                         [9] Protection—Control,
                                         [10] Display—Control,
                                         [11] Device—Control,
                                         [12] Cursor—Control,
                                         [13] Reset—Control }
```

This subsection addresses the additional format effector characters which must be specially handled in conversion between the various data syntaxes.

A repeat function is available in all of the data syntaxes; however, the side effects of the function differ between the data syntaxes. Terminal data syntax DS I provides a function which allows the immediately preceding G—set character, or pair of characters in the case of a composite coded graphic character and non spacing accent character, to be repeated. Both terminal data syntaxes DS II and DS III restrict the character to a graphic character (i.e. an alphanumeric text character or mosaic character from the repertoire, or a DRCS character). These limitations must be considered in establishing the conversion process. Here the repeat function will be considered to repeat any preceding G—set character and testing must be performed in the interpretation of the IDS to eliminate any erroneous cases.

The functions hold mosaic and release mosaic occur in only one data syntax and require special interpretation since analogous functions do not exist directly in any of the terminal data syntaxes.

A.3.9.1 *Other format effectors*

```
Other—Format—Effectors ::= CHOICE { [1] Repeat—N,
                                     [2] Repeat—EOL,
                                     [3] Hold—Mosaic,
                                     [4] Release—Mosaic }
```

A.3.9.1.1 *Repeat—N*

Repeat—N ::= SEQUENCE { SID, RPT—Par }

—— Special character indicating the REPEAT Function from Recommendation T.101 DS I [C1 set position 5/8, (9/8)], DS II [C0 set position 1/2), DS III [C1 set position 4/6, (8/6)].

RPT—Par ::= INTEGER

—— Count of repetitions.

A.3.9.1.2 *Hold mosaic*

Repeat—EOL ::= SID

—— Special character indicating the REPEAT TO End Of Line Function from Recommendation T.101 DS I [C1 set position 5/8, (9/8)] with parameter 0), DS III [C1 set position 4/7, (8/7)].

A.3.9.1.3 *Hold mosaic*

Hold—Mosaic ::= SID

—— Special character indicating the Hold—Mosaic function (HMS) from Recommendation T.101 DS II [C1 set (serial) position 5/14, (9/14)].

A.3.9.1.4 *Release mosaic*

Release—Mosaic ::= SID

—— Special character indicating the Release—Mosaic (RMS) function from Recommendation T.101 DS II [C1 set (serial) position 5/15, (9/15)].

A.3.9.2 *Lining control*

The lining function permits an underline to be displayed as part of the graphics character shape for alphanumeric characters from Repertoire 1. This underline is considered as a part of the character cell image before any rotation operation is applied. In the special case of the display of mosaic characters, the lining function establishes a "separated mosaic" font. The capability to handle separated mosaics is available in all of the three terminal data syntaxes; however, the level of capability differs. In terminal data syntax DS II, only one size of separation for separated mosaics is directly available. In terminal data syntaxes DS I and DS III the amount of separation is defined by the line width (drawing point size) parameter (logical pel) in the geometric drawing commands. Basic separated mosaics may be converted directly between each of the data syntaxes. Since the variation in separation cannot be achieved directly in one of the terminal data syntax it must be simulated by the use of DRCS. Of course simulation of separated mosaics in this manner would consume limited DRCS resources and therefore must consider the boundary condition specification.

Lining—Control ::= INTEGER { start—lining (1),
stop—lining (2) }

—— Start—Lining is a function from Recommendation T.101 DS I and II [C1 set position 5/10, (9/10)] and (UNDERLINE START) from DS III [C1 set position 5/9, (9/9)].

—— Stop—Lining is a function from Recommendation T.101 DS I and II [C1 set position 5/9, (9/9)] and (UNDERLINE STOP) from DS III [C1 set position 5/10, (9/10)].

A.3.9.3 *Character size control*

The various terminal data syntaxes provide the capability to establish a wide range of character sizes for basic alphanumeric text, mosaics and DRCS characters. In addition, terminal data syntax DS II provides the capability to separately define completely variable character sizes for text defined as part of the geometric part of DS II. Since this "geometric text" data is used only for the annotation of geometric pictures in the optional geometric part of DS II, it is not necessary to consider it as part of the translation of basic alphanumeric text. DS III, on the other hand, provides only one form of text. Therefore it is necessary to handle operations such as dynamic text sizes and rotations as part of the conversion between data syntaxes.

Since the capability to scale text, mosaics and DRCS to arbitrary sizes is not available in all data syntaxes, there will be some degradation of the displayed image when converted from one data syntax to another. It is undesirable to lose any textual information in the conversion process, since this textual information might be of principal importance to the understanding of the videotex page. Also it is not desirable to arbitrarily wrap or scroll textual information since this would corrupt mosaic information. In certain situations the conversion process must automatically choose a smaller size of character cell in order to avoid the loss of information. The commands for character size control indicate the size of the character cell intended in the terminal data syntax used to represent the source data. The resultant character cell in the converted form may be smaller depending upon the capabilities of the target terminal data syntax and the boundary condition in effect.

Two separate functions exist to define characters of double height. This is due to a difference in the definition of the relationship of the double height character cell to the location of the baseline in part of one of the source data syntaxes. Since data syntax DS II provides a capability to define double height characters which both extend up a double height above the baseline, and which extend down below the baseline, two functions are provided here. Since the other two terminal data syntaxes provide only a single double height capability, a conversion involving a repositioning of the baseline is required.

```
Character—Size—Control ::= CHOICE { [1] Normal—Size,
                                    [2] Double—Size—Up,
                                    [3] Double—Width,
                                    [4] Double—Height—Up,
                                    [5] Double—Height—Down,
                                    [6] Small—Size,
                                    [7] Medium—Size,
                                    [8] Double—Size—Down }
```

A.3.9.3.1 *Normal size*

```
Normal—Size      ::= SID
```

—— A function from Recommendation T.101 DS I [C1 set position 4/10, (8/10)], from DS II [C1 set position 4/12, (8/12)] and (NORMAL TEXT) from DS III [C1 set position 4/12, (8/12)].

—— *Note* — The "Normal—Size" of text is defined by the boundary conditions of each of the terminal data syntaxes and is not the same in any of the terminal data syntaxes. Although the width of the "normal" character cell size is 1/40 of the screen width in DS II and DS III, the screen width is not exactly the same. The "normal" character cell size in DS I is by default 1/31 of the width; however, it may be redefined by the DS I P—TEXT command. Similarly the vertical height of a character cell differs between the various terminal data syntaxes. This command indicates that the source terminal data syntax intended to use "normal" size implicit in that terminal data syntax. This will require conversion to the normal size implicit in that resultant terminal data syntax. The value of "normal size" should be communicated explicitly in the state vector associated with this command.

A.3.9.3.2 *Double size up*

```
Double—Size—Up       ::= SID
```

—— A function from Recommendation T.101 DS II [C1 set position 4/15, (8/15)], (DOUBLE SIZE) from DS III [C1 set position 4/15, (8/15)], and (DBS 4/5) from DS I [C1 set position 4/11, (8/11) followed by parameter 4/5].

—— *Note* — The character cell width and height are twice that defined by the control command "Normal—Size".

A.3.9.3.3 *Double width*

```
Double—Width         ::= SID
```

—— A function from Recommendation T.101 DS II [C1 set position 4/14, (8/14)], and (DBW 4/4) from DS I [C1 set position 4/11, (8/11) followed by parameter 4/4].

—— *Note* — The character cell width is twice that defined by the control command "Normal—Size".

A.3.9.3.4 *Double height up*

```
Double—Height—Up     ::= SID
```

—— A function from Recommendation T.101 DS II [C1 set position 4/13, (8/13)], (DOUBLE HEIGHT) from DS III [C1 set position 4/13, (8/13)], and (DBH 4/1) from DS I [C1 set position 4/11, (8/11) followed by parameter 4/1].

—— *Note* — The character cell height is twice that defined by the control command "Normal—Size" and extends up two character cell heights above the baseline.

A.3.9.3.5 *Double height down*

Double—Height—Down ::= SID

—— A function from Recommendation T.101 DS II [C1 set position 4/13, (8/13)].

—— *Note* — The character cell height is twice that defined by the control command "Normal—Size" and the double height character extends one cell height above the baseline and one cell height below the baseline.

A.3.9.3.6 *Small size*

Small—Size ::= SID

—— A function from Recommendation T.101 DS I [C1 set position 4/8, (8/8)] and (SMALL TEXT) from DS III [C1 set position 4/10, (8/10)].

—— *Note* — The character cell width and height are half that defined by the control command "Normal—Size".

A.3.9.3.7 *Medium size*

Medium—Size ::= SID

—— A function from Recommendation T.101 DS I [C1 set position 4/9, (8/9)] and (MEDIUM TEXT) from DS III [C1 set position 4/11, (8/11)].

—— *Note* — The character cell size is defined to be an intermediate size. This intermediate size is defined by the boundary conditions of each of the source data syntaxes which use this control function. In data originating from data syntax DS III, medium size is defined to be 1/32 the normalized width of the display area and 3/64 the height of the normalized unit area. In data from data syntax DS I, medium text becomes half the character cell height and the full width defined by the control command "Normal—Size".

A.3.9.3.8 *Double size down*

Double—Size—Down ::= SID

—— A function from Recommendation T.101 DS II [C1 set position 4/15, (8/15)].

A.3.9.4 *Flash control*

The operation of the flash capability is dependent on the terminal model of the particular source data syntax. In a "multi—plane" terminal configuration, character cells may have an implicit foreground and background which alternate during blinking. In a "single—plane" terminal configuration, the flash capability is achieved by use of colour mapping operations. It is possible to convert between these two variants on flashing. In addition to a basic flash capability driven by control characters, each of the terminal data syntaxes also provides the capability to establish complex dynamic important to reference the boundary condition imposed by the number of colours in the colour map and the terminal model plane structure.

```
Flash—Control    ::= SEQUENCE { Flash—Rate, Flash—Mode }

Flash—Rate       ::= CHOICE { [1] Flash,
                              [2] Steady,
                              [3] Phase1—Flash,
                              [4] Phase2—Flash,
                              [5] Phase3—Flash,
                              [6] Increment—Flash,
                              [7] Decrement—Flash,
                              [8] Blink—Stop }

Flash—Mode       ::= CHOICE { [1] Normal,
                              [2] Inverted—Flash,
                              [3] Reduced—Intensity—Flash }
```

A.3.9.4.1 *Flash*

Flash ::= SID

—— A function from Recommendation T.101 DS II [C1 set position 4/8, (8/8)], (FLC 4/0) from DS I [C1 set position 5/1, (9/1)] followed by parameter 4/0] and (BLINK START) from DS III [C1 set position 4/14, (8/14)].

—— *Note* — Establish a 50% cycle flash either from the foreground to the background or between two colour map addresses chosen implicitly to produce the equivalent effect of foreground/background flashing. Although the Flash function is similar in the three source data syntaxes, the rate of flashing is not necessarily the same.

A.3.9.4.2 *Steady*

Steady ::= SID

—— A function from Recommendation T.101 DS II [C1 set position 4/9, (8/9)], (FLC 4/15) from DS I [C1 set position 5/1, (9/1)] followed by parameter 4/15].

—— *Note* — Cancel the application of any flashing attribute.

A.3.9.4.3 *Invested flash*

Inverted—Flash ::= SID

—— A function from Recommendation T.101 DS II (C1 set position CSI 3/0 4/1), (FLC 4/7) from DS I [C1 set position 5/1, (9/1)] followed by parameter 4/7].

—— *Note* — Establish an inverted phase 50% cycle flash from the foreground to the background.

A.3.9.4.4 *Reduced intensity flash*

Reduced—Intensity—Flash ::= SID

—— A function from Recommendation T.101 DS II (C1 set position CSI 3/1 4/1), (FLC 4/7) from DS I [C1 set position 5/1, (9/1)] followed by parameter 4/7].

—— *Note* — Establish a reduced intensity flash between colour map addresses.

A.3.9.4.5 *Phase 1—flash*

Phase 1—Flash ::= SID

—— A function from Recommendation T.101 DS II (C1 set position CSI 3/2 4/1), (FLC 4/4) from DS I [C1 set position 5/1, (9/1)] followed by parameter 4/4].

—— *Note* — Establish a 33% cycle flash from the foreground to the background beginning on phase 1.

A.3.9.4.6 *Phase 2—flash*

Phase 2—Flash ::= SID

—— A function from Recommendation T.101 DS II (C1 set position CSI 3/3 4/1), (FLC 4/2) from DS I [C1 set position 5/1, (9/1)] followed by parameter 4/2].

—— *Note* — Establish a 33% cycle flash from the foreground to the background beginning on phase 2.

A.3.9.4.7 *Phase 3—flash*

Phase 3—Flash ::= SID

—— A function from Recommendation T.101 DS II (C1 set position CSI 3/4 4/1), (FLC 4/1) from DS I [C1 set position 5/1, (9/1)] followed by parameter 4/1].

—— *Note* — Establish a 33% cycle flash from the foreground to the background beginning on phase 3.

A.3.9.4.8 *Increment flash*

Increment—Flash ::= SID

—— A function from Recommendation T.101 DS II (C1 set position CSI 3/5 4/1).

—— *Note* — Establish a 33% cycle flash from the foreground to the background incrementing the phase reference.

A.3.9.4.9 *Decrement flash*

Decrement—Flash ::= SID

—— A function from Recommendation T.101 DS II (C1 set position CSI 3/6 4/1).

—— *Note* — Establish a 33% cycle flash from the foreground to the background incrementing the phase reference.

A.3.9.4.10 *Blink stop*

Blink—Stop ::= SID

—— A function from Recommendation T.101 DS III [C1 set position 5/14, (9/14)].

—— *Note* — Stop all blink processes.

A.3.9.5 *Conceal control*

The conceal display function is intended for operation on a terminal model which supports multiple independent planes. Data stored in character cells may be marked as concealed, in which case the background of the character cell will display in the same colour as the background of the cell. A local reveal command would cause the foreground to be displayed in the originally defined colours. A conversion is necessary to handle this function on a single plane terminal. The capability may be simulated either by the use of a key activated macro which contains a definition of the foreground of the concealed character cells or it may be simulated by the colour map. The definition of the key activated macro sequence must be established during a sorting process in the conversion procedure and is limited by the availability of macro memory. Use of the colour map for the simulation of this function consumes colour map resources very quickly. Therefore the handling of the conceal function should be the lowest priority in using colour map resources. The conceal and stop conceal control functions are included here so that they may be handled in the most effective manner by the conversion process.

Conceal—Control ::= CHOICE { [1] Conceal—Display,
[2] Stop—Conceal—Display }

A.3.9.5.1 *Conceal display*

Conceal—Display ::= SID

—— A function from Recommendation T.101 DS II [C1 set position 5/8, (9/8)] and DS I [C1 set position 5/2, (9/2)] followed by parameter 4/0].

—— *Note* — Establish a Conceal state attribute.

A.3.9.5.2 *Stop conceal display*

Stop—Conceal—Display ::= SID

—— A function from Recommendation T.101 DS II (C1 set position CSI 4/2) and DS I [C1 set position 5/2, (9/2) followed by parameter 4/15].

—— *Note* — Stop applying Conceal state attribute.

A.3.9.6 *Invert control*

Invert—Control ::= CHOICE { [1] Invert—Polarity,
[2] Normal—Polarity }

—— *Note* — Invert the application of the foreground and background colour attributes in a multi—plane terminal model environment and invert the overlaying (foreground) and underlaying (background) colours in a single plane terminal environment. These commands have essentially the same effect when generating a presentation in each of the identified terminal model environments; however, there is a great difference in the effect when this command is used to change the attributes of an already displayed graphic character. This must be handled in the conversion by the process which converts the effects of different planes of the terminal model.

A.3.9.6.1 *Invert polarity*

Invert—Polarity ::= SID

—— A function from Recommendation T.101 DS II [C1 set position 5/12, (9/12)] and (REVERSE VIDEO) DS III [C1 set position 4/8, (8/8)].

—— *Note* — Establishes Invert Polarity attribute.

A.3.9.6.2 *Normal polarity*

Normal—Polarity ::= SID

—— A function from Recommendation T.101 DS II [C1 set position 5/13, (9/13)] and (NORMAL VIDEO) DS III [C1 set position 4/9, (8/9)].

—— *Note* — Establishes Normal Polarity attribute.

A.3.9.7 *Window/box control*

The window/box capability establishes a special background colour for a character cell which is transparent to a video image which may underlay the display. This capability is provided directly by two control commands in one of the source terminal data syntaxes. The same capability is provided in a more complex manner in all of the data syntaxes by the establishment of a special transparent colour which may be used together with other presentation commands.

```
Window/Box—Control ::= INTEGER { start—box   (1),
                                 end—box     (2) }
```

—— Start—Box is a function from Recommendation T.101 DS II [C1 set position 4/10, (8/10)].

—— *Note* — Establish the Boxing attribute.

—— End—Box is a function from Recommendation T.101 DS II [C1 set position 4/11, (8/11)].

—— *Note* — Stop applying the Boxing attribute.

A.3.9.8 *Marking control*

The marking control capability marks character cell locations for further action. This function depends upon the availability of a character cell—oriented memory in the terminal model. It cannot be converted to other data syntaxes.

```
Marking—Control      ::= INTEGER { marked—mode—start      (1),
                                   marked—mode—stop       (2) }
```

—— Marked—Mode—Start is a function from Recommendation T.101 DS II (C1 set position CSI 3/0 5/3, CSI 3/1 5/3 or CSI 3/2 5/3).

—— *Note* — Apply the Marking attribute.

—— Marked—Mode—Stop is a function from Recommendation T.101 DS II (C1 set position CSI 3/0 5/4, CSI 3/1 5/4 or CSI 3/2 5/4).

—— *Note* — Stop applying Marking attribute.

A.3.9.9 *Protection control*

The manner in which selective input control is handled in the three source terminal data syntaxes differs greatly. Not only are the procedures different but the input processes are bounded by different boundary conditions. For example, in one case input is associated with the character cell memory of the multi—plane terminal model, whereas in another case, such input data is bounded by a storage limit on the number and cumulative size of such input fields. Since such input processes are fundamentally different, the commands which control them are included here separately. This will permit the conversion process to simulate one set of functions in a different terminal environment.

```
Protection—Control ::= INTEGER { unprotect—field      (1),
                                 protect—field        (2),
                                 protect—mode—start   (3),
                                 protect—mode—cancel  (4),
                                 protect—mode—idle    (5),
                                 unprotect—block      (6),
                                 protect—block        (7) }
```

—— Unprotect—Field is a function from Recommendation T.101 DS III [C1 set position 5/15, (9/15)].

—— *Note* — Unprotect a given area of the display screen, defined by the FIELD geometric command, to allow the input of characters into the unprotected field buffer when the cursor is in the unprotected area.

—— Protect—Field is a function from Recommendation T.101 DS III [C1 set position 5/0, (9/0)].

—— *Note* — Protect a given area of the display screen to prevent the input of characters into the unprotected field buffer when the cursor is in the unprotected area. The entire screen area is protected by default.

—— Protect—Mode—Start is a function from Recommendation T.101 DS II (C1 set position CSI 3/0 5/0, CSI 3/1 5/0 or CSI 3/2 5/0).

—— *Note* — Apply the protected attribute to character cell positions preventing overwriting.

—— Protect—Mode—Cancel is a function from Recommendation T.101 DS II (C1 set position CSI 3/0 5/1, CSI 3/1 5/1 or CSI 3/2 5/1).

—— *Note* — Cancel the protected attribute to character cell positions allowing overwriting.

—— Protect—Mode—Idle is a function from Recommendation T.101 DS II (C1 set position CSI 3/2 5/2).

—— *Note* — Stop the application of the protect mode attribute.

—— Unprotect—Block is a function from Recommendation T.101 DS I [C1 set position 5/14, (9/14)].

—— *Note* — Remove protection of character cell positions against alteration.

—— Protect—Block is a function from Recommendation T.101 DS I [C1 set position 5/15, (9/15)].

—— *Note* — Protect character cell positions against alteration.

A.3.9.10 *Display control*

The display control subcategory of commands contains functions which affect the manner in which the display device presents information. This includes configuration of the display memory available in a particular terminal model, includes whether the contents of that display memory is to be scrolled, and includes the overwriting of information in the display memory.

```
Display—Control ::= CHOICE { [1] Plane—Configuration—Control,
                             [2] Scroll—Control,
                             [3] Overwrite—Mode }
```

The terminal models used in each of the three terminal data syntaxes differ significantly from one another. In two of the cases the terminal model structures are fixed. In the case of data syntax DS I the terminal model structure can be altered dynamically. The amount of display memory assigned to each display plane and the presentation (overlay) order of the planes may be altered. These functions are highly dependent upon the display hardware used to realize the particular display model which underlies data syntax DS I and the dynamic effects which may be generated using these functions cannot be converted to either of the other data syntaxes. However these comamnds must be interpreted by the conversion process in order to establish the criteria for sorting other display information to achieve the mapping from the data syntax DS I multi plane terminal model to the different data syntax DS II multi—plane terminal model or to the data syntax DS III single plane terminal model.

A.3.9.10.1 *Plane configuration control*

```
Plane—Configuration—Control ::= CHOICE { [1] Frame—Area,
                                          [2] Set—Frame,
                                          [3] Assign—Frame,
                                          [4] Header—Area,
                                          [5] Body—Area }
```

A.3.9.10.1.1 *Frame area*

```
Frame—Area ::= SEQUENCE { Area—Origin, Area—Dimensions }
```

—— The Frame Area Function is from Recommendation T.101 DS I [Display Control command set position 2/5, (10/5)].

—— *Note* — The Display Control Command G Set has final character 3/8 within DS I.

```
Area—Origin      ::= SEQUENCE { REAL, REAL }
```

—— Specification of the origin of the Frame Area.

Area—Dimensions ::= SEQUENCE { REAL, REAL }

—— Specification of the dimensions of the Frame Area.

—— Coordinates are specified as normalized fractions of the unit screen area represented in a signed integer field with an implied binary point in the most significant place.

A.3.9.10.1.2 *Set frame*

Set—Frame ::= SEQUENCE OF { Set—Frame—Index,
Set—Frame—Memory—Assignment }

—— The Frame Area Function is from Recommendation T.101 DS I [Display Control command set position 2/6, (10/6)].

Set—Frame—Index ::= INTEGER

—— Frame Area Index.

Set—Frame—Dimensions ::= INTEGER

—— Number of bits of raster memory allocated to the frame.

A.3.9.10.1.3 *Assign frame*

Assign—Frame ::= INTEGER

—— A function from Recommendation T.101 DS I [Display Control command set position 2/7, (10/7)].

A.3.9.10.1.4 *Header area*

Some of the terminal oriented Videotex data syntaxes provide a capability to present information in a special message area as well as in the main display area. This message area would contain service oriented messages. The content of these messages would doubtless change in international interworking between Videotex systems. Data syntax DS I provides special commands which control this message header. In DS I the raster and header raster commands control the display of presentation information in the main display area or the header message area. The raster commands also establish the initial colour values in data syntax DS I. These commands are included here so that header information can be identified and properly converted.

Header—Area ::= SEQUENCE { Raster—Colour—Value }

—— A function from Recommendation T.101 DS I [Display Control command set position 3/9, (11/9)].

—— *Note* — The Display Control Command G Set has final character 3/8 within DS I.

Raster—Colour—Values ::= SEQUENCE { INTEGER, INTEGER, INTEGER }

—— Specification of the initial raster header colour for Green, Red, and Blue respectively.

—— Colour values are specified as normalized fractions of the unit range of colours represented in a signed integer field with an implied binary point in the most significant place.

A.3.9.10.1.5 *Body area*

Body—Area ::= SEQUENCE { Body—Opcode, Raster—Colour—Values }

—— A function from Recommendation T.101 DS I [Display Control command set position 3/8, (11/8)].

A.3.9.10.2 *Scroll control*

Scrolling may occur on a whole screen basis or on a partial screen basis. There is a major difference between scrolling on a multi—plane terminal model and on a single plane terminal model. As well the assignment of functions to the various planes in a multi—plane terminal model also makes a tremendous difference to the result of scrolling. In some cases the underlying graphics information moves with the scrolling characters and in other cases it remains in place. In DS I the multi—plane motion capability permits dynamic motions and plane assignments which greatly affect how scrolling operates. In general it is not possible to convert all dynamic operations such as scrolling between terminal data syntaxes; however, the results of scrolling affects the final presentation. The conversion process must buffer data and post—process it so that the final image is correct. Since the scroll operations in each of the three terminal data syntaxes are fundamentally different, they are all included here so that the conversion process can handle them.

Scroll—Control ::= CHOICE { scroll—on [1] NULL,

```
scroll—off                          [2] NULL,
scroll—up                           [3] NULL,
scroll—down                         [4] NULL,
activate—implicit—scrolling         [5] NULL,
deactivate—implicit—scrolling       [6] NULL,
create—scroll—area                  [7] Create—Scroll—Area,
delete—scroll—area                  [8] Delete—Scroll—Area,
scroll—display—mode—on              [9] NULL,
scroll—display—mode—off             [10] NULL }
```

—— Scroll—On is a function from Recommendation T.101 DS III [C1 set position 5/7, (9/7)].

—— *Note* — Enable single plane scroll within an active display Field.

—— Scroll—Off is a function from Recommendation T.101 DS III [C1 set position 5/8, (9/8)].

—— *Note* — Disable single plane scroll.

—— Scroll—Up is a function from Recommendation T.101 DS II (C1 set position CSI 3/0 6/0).

—— *Note* — Cause the scrolling area to scroll up.

—— Scroll—Down is a function from Recommendation T.101 DS II (C1 set position CSI 3/1 6/0).

—— *Note* — Cause the scrolling area to scroll down.

—— Activate—Implicit—Scrolling is a function from Recommendation T.101 DS II (C1 set position CSI 3/2 6/0).

—— *Note* — Cause the scrolling area to scroll implicitly on encountering scroll area boundary.

—— Deactivate—Implicit—Scrolling is a function from Recommendation T.101 DS II (C1 set position CSI 3/3 6/0).

—— *Note* — Cause the scrolling area not to scroll implicitly.

—— Scroll—Display—Mode—On is a function from Recommendation T.101 DS I (Display Control Command G Set position 2/4 with parameter b6 = 1).

—— *Note* — The Display Control Command G Set has final character 3/8 within DS I.

—— *Note* — Establish the Scroll attribute of the Display Mode.

—— Scroll—Display—Mode—Off is a function from Recommendation T.101 DS I (Display Control Command G Set position 2/4 with parameter b6 = 0).

—— *Note* — Disable the Scroll attribute of the Display Mode.

A.3.9.10.2.1 *Create scroll area*

```
Create—Scroll—Area ::= SEQUENCE { Upper—Par, Lower—Par }
```

—— A function from Recommendation T.101 DS II (5/5).

—— *Note* — Create a scrolling area.

```
Upper—Par       ::= SEQUENCE { INTEGER, INTEGER, INTEGER }
```

—— Parameters <URH> <URT> <URU> defining the upper boundary row of the scrolling area.

```
Lower—Par       ::= SEQUENCE { INTEGER, INTEGER, INTEGER }
```

—— Parameters <LRH> <LRT> <LRU> defining the lower boundary row of the scrolling area.

A.3.9.10.2.2 *Delete scroll area*

```
Delete—Scroll—Area  ::= SEQUENCE { Upper—Par, Lower—Par }
```

—— A function from Recommendation T.101 DS II (5/6).

—— *Note* — Delete a scrolling area.

A.3.9.10.3 *Overwrite mode*

In conjunction with control over the terminal model memory configuration, one of the terminal data syntaxes provides a unique capability of controlling how data builds up in a particular display plane. Data syntax DS I allows the overwriting of memory to be dependent upon the current contents of memory. The new data may either replace the old contents of memory, or perform a logical "OR", logical "AND", or logical "XOR" (eXclusive OR) with the old contents of the memory before replacing it. This function is extremely difficult to simulate in either of the other two data syntaxes in the general case since it requires operations at the bit level within a particular terminal model dependent memory. It is included here so that the conversion process can perform the best simulation possible.

Overwrite—Mode ::= SEQUENCE { Overwrite—Par }

—— A function from Recommendation T.101 DS I (Display Control Command G Set position 2/4).

—— *Note* — The Display Control Command G Set has final character 3/8 within DS I.

```
Overwrite—Par      ::= INTEGER { replace      (1),
                                 or           (2),
                                 and          (3),
                                 xor          (4) }
```

A.3.9.11 *Device control*

Except for the display device on or off commands, the device—control commands control other than presentation display functions and are outside the scope of the interworking data syntax.

```
Device—Control ::= INTEGER { display—device—on      (1),
                             display—device—off     (2) }
```

—— Display—Device—On is a function from Recommendation T.101 DS II (Control Sequence ESC 3/12).

—— Display—Device—Off is a function from Recommendation T.101 DS II (Control Sequence ESC 3/13).

A.3.9.12 *Cursor control*

The display cursor is controlled explicitly in each of the terminal data syntaxes as well as implicitly in one. In addition the explicit cursor control commands do not have the same coding in any of the terminal data syntaxes. In the implicit case of cursor control, the display cursor is controlled by the unprotected field protect mode control in terminal data syntax DS III. Conversion is required between each of these control functions.

```
Cursor—Control  ::= CHOICE { Cursor—On     (1),
                             Cursor—Flash  (2),
                             Cursor—Off    (3) }
```

A.3.9.12.1 *Cursor—on*

Cursor—On ::= SID

—— A function from Recommendation T.101 DS II (C0 set position 1/1), DS I [C1 set position 4/14, (8/14)] and (CURSOR STEADY) from DS III [C1 set position 5/12, (9/12)].

A.3.9.12.2 *Cursor flash*

Cursor—Flash ::= SID

—— A function from Recommendation T.101 DS III [C1 set position 5/11, (9/11)].

A.3.9.12.3 *Cursor—off*

Cursor—Off ::= SID

—— A function from Recommendation T.101 DS II (C0 set position 1/14), DS I [C1 set position 4/15, (8/15)] and (CURSOR OFF) from DS III [C1 set position 5/13, (9/13)].

A.3.9.13 *Reset control*

Each of the source Videotex data syntaxes provides the capability to reset the states of the display environment supporting that particular data syntax to a predefined set of values. The parameters which may be altered by the various reset functions in the different source data syntaxes are quite different. Some of the reset functions provide the capability to reset particular parameters selectively while others reset a data syntax dependent predefined list of parameters. The interworking data syntax must support reset functions in two different ways. Firstly an indication of the particular reset command must be communicated as a syntactic element within the IDS. The various reset functions are included here so that the presentation affect of the reset function may be effected in the conversion. Secondly a reset function greatly effects the global presentation states. These states are kept track of in the conversion process so that the conversion process need not understand the interrelationship between presentation commands. This means that the conversion process does not have to simulate a terminal of the source data syntax in order to handle the conversion of its elements. Therefore along with an IDS reset control command it is necessary to include a special form of the state vector which re—establishes the global variables.

```
Reset—Control     ::= CHOICE { [1] Reset—Type—I,
                               [2] Reset—Type—II,
                               [3] Reset—Type—III }
```

A.3.9.13.1 *Reset type—I*

```
Reset—Type—I      ::= SEQUENCE { P—Reset—Par OPTIONAL }
```

—— A function from Recommendation T.101 DS I [Display Control Command G Set position 2/1, (10/1)].

—— *Note* — The Display Control Command G Set has final character 3/8 within DS I.

A.3.9.13.1.1 *P—reset par*

```
P—Reset—Par       ::= SEQUENCE { macro—reset BOOLEAN,
                                 blink—reset BOOLEAN,
                                 lut—reset BOOLEAN,
                                 screen—reset BOOLEAN }
```

—— Selectively reset the identified parameters.

—— *Note* — Data Syntax DS I also includes the NSR reset function which is identified separately above.

A.3.9.13.2 *Reset type—II*

```
Reset—Type—II     ::= SEQUENCE { US—Reset—Operation,
                                 US—Reset—Parameter }
```

—— A function from Recommendation T.101 DS II (C0 set position 1/15) followed by fixed character 2/15.

A.3.9.13.2.1 *US—reset operation*

```
US—Reset—Operation ::= CHOICE { us—reset—mosaic—1              [1] NULL,
                                us—reset—mosaic—2              [2] NULL,
                                us—reset—mosaic—1—limited      [3] NULL,
                                 us—reset—mosaic—2—limited     [4] NULL,
                                us—reset—service—break         [5] US—Reset—Service—Break,
                                us—reset—to—previous—state     [6] NULL }
```

—— US—Reset—Mosaic—1 is represented by US Reset Identifier Character (4/1), and resets to defaults and invokes serial C1 set.

—— US—Reset—Mosaic—2 is represented by US Reset Identifier Character (4/2), and resets to defaults and invokes parallel C1 set.

—— US—Reset—Mosaic—1—Limited is represented by US Reset Identifier Character (4/3), and resets to limited defaults and invokes parallel C1 set.

—— US—Reset—Mosaic—2—Limited is represented by US Reset Identifier Character (4/4), and resets to limited defaults and invokes parallel C1 set.

—— US—Reset—to—Previous—State is represented by US Reset Identifier Character (4/15), and resets to previous state after a reset to service break.

A.3.9.13.2.2 *US—reset service break*

```
US—Reset—Service—Break ::= SEQUENCE { INTEGER { break—to—row—serial (1),
                                                 break—to—row—parallel (2) }, row—designator }
```

—— Break—to—Row—Serial is represented by US Reset Identifier Character (4/0), and service breaks to row serial C1 set.

—— Break—to—Row—Parallel is represented by US Reset Identifier Character (4/5), and service breaks to row parallel C1 set.

—— Row—Designator is represented by US Reset Row Designator Parameter Character, where the designated row is coded from columns 4 to 7 of the code table. The row number is indicated by the binary value of the 6 least significants bits.

A.3.9.13.3 *Reset type—III*

```
Reset—Type—III      ::= SEQUENCE { [1] Reset—Par1 OPTIONAL,
                                   [2] Reset—Par2 OPTIONAL }
```

—— A function from Recommendation T.101 DS III [PD1 G Set position 2/0, (10/0)].

```
Reset—Par1          ::= SEQUENCE { INTEGER {
                                       colour—mode—1                                  (1),
                                       colour—mode—2                                  (2),
                                       colour—mode—3                                  (3), }
                                   {INTEGER {
                                       display—to—nominal—black                  (1),
                                       display—to—current—colour                      (2),
                                       border—to—nominal—black                        (3),
                                       border—to—current—colour                       (4),
                                       display—and—border—to—current—colour           (5),
                                       display—to—current—colour—and—border
                                       —to—nominal—black                         (6),
                                       display—and—border—to—nominal—black            (7) },
                                   domain BOOLEAN }

Reset—Par2          ::= SEQUENCE {
                                   drcs—reset BOOLEAN,
                                   macro—pdi—reset BOOLEAN,
                                   texture—reset BOOLEAN,
                                   unprotected—field—reset BOOLEAN,
                                   blink—pdi—reset BOOLEAN,
                                   text—pdi—reset BOOLEAN }
```

—— Selectively reset the identified parameters.

—— *Note* — Data Syntax DS III also includes the NSR reset function which is identified separately above.

A.3.10 *Geometric string*

All of the source terminal data syntaxes provide a geometric capability, however the capabilities available in each of the geometric systems is quite different. The interworking data syntax groups the common geometric commands together. Recommendation F.300 has identified the various categories into which geometric functions may be organized. These categories are used below. The IDS uses normalized coordinates for all geometric commands. Also the IDS uses relative coordinate specifications for all lists of coordinates, except for the set—position and marker—point commands which are absolute. However, in certain cases there may be a choice of either absolute or relative coordinates. All other forms of coordinates used within any of the terminal oriented data syntaxes, such as the general use of absolute or incremental, will be converted to the forms indicated above.

```
Geometric—String    ::= CHOICE { [1] Geometric—Drawing—Command,
                                 [2] Geometric—Control—Command }
```

A.3.10.1 *Geometric drawing command*

```
Geometric—Drawing—Command ::= CHOICE { [1] Marker—Point,
                                        [2] Line,
                                        [3] Arc—Circle,
                                        [4] Rectangle,
                                        [5] Polygon,
                                        [6] Spline,
                                        [7] Pixel—Array }
```

Some of the source terminal data syntaxes provide a method of optionally carrying on from one primitive to another in a relative manner. This provides a level of efficiency in certain situations, however the multiplicity of equivalent formats would make the interworking data syntax more complex. Therefore the interworking data syntax requires the specification of the initial position of a drawing command as part of the string of parameters for each command. In some situations in converting from data syntaxes which permit the relative association of commands it will be necessary for the conversion process to calculate the current position in effect at the beginning of a command and include that data as part of the parameter string. There is no direct equivalent in the IDS of the data syntaxes I and III set position command. This information is carried as the initial parameter of each of the other drawing commands.

A.3.10.1.1 *Marker point*

The various terminal data syntaxes differ in their capability to present a marker shape at a point. Data syntaxes DS I and DS III provide only the capability to draw a dot, whereas data syntax DS II also provides the capability to draw a marker shape at a specific point. The conversion process can easily simulate the marker point functionality in converting to data syntax DS I or DS III by the use of more than one presentation function, possibly included in a MACRO command for efficiency. The dot—point or shape—point command is identified by the context tag in the CHOICE statement. The shape of the shape point (marker) is defined by a geometric control command.

```
Marker—Point        ::= CHOICE { [1] Dot—Point,
                                 [2] Shape—Point }

Dot—Point    ::= SEQUENCE OF { Abs—Coord }
```

—— This command carries the functionality of the Data Syntax I, and III SET POINT command and of the Data Syntax II POLYMARKER command, with the marker the shape of a dot.

```
Shape—Point      ::= SEQUENCE OF { Abs—Coord }
```

—— This command carries the functionality of the Data Syntax II POLYMARKER command with a general marker shape.

A.3.10.1.2 *Line*

All of the terminal data syntaxes provide the capability to draw a single or a series of lines. Minor differences exist with respect to the manner in which boundary conditions are handled, however in general a direct conversion is possible.

```
Line ::= SEQUENCE OF { Abs—Coord, SEQUENCE OF { Rel—Coord }
```

—— This command carries the functionality of the Data Syntax I, and III LINE command and of the Data Syntax II POLYLINE command.

A.3.10.1.3 *Arc—circle*

The capability to draw an arc or a circle differs somewhat between the various data syntaxes. In each of the various data syntaxes the circle/arc function has been optimized to such an extent that it provides an efficient manner of communicating arc or circle information in the context of that data syntax. The interworking data syntax is less concerned about efficiency that it is about carrying sufficient information to permit conversion to take place. Therefore alternate ways of carrying the same parameters will not be addressed by the IDS, however the IDS will include all the functions available in the circle—arc capability in the various data syntaxes.

```
Arc—Circle ::= CHOICE { [1] Circle,
                        [2] Arc—3—Point,
                        [3] Arc—3—Point—Chord,
                        [4] Arc—3—Point—Pie,
                        [5] Ellipse,
                        [6] Elliptic—Arc,
```

[7] Elliptic—Arc—Chord,
[8] Elliptic—Arc—Pie,
[9] Arc—Centre—Cord,
[10] Arc—Centre—Pie }

A.3.10.1.3.1 *Circle*

Circle ::= SEQUENCE { Abs—Coord, Coord }

—— This command carries the functionality of the Data Syntax I, and III ARC command (circle form) and of the Data Syntax II GDP (circle) command.

—— The absolute coordinate defines the initial position of the circle. The other coordinate defines the diameter of the circle by specifying a point on the opposite side.

A.3.10.1.3.2 *Arc—3 point*

Arc—3—Point ::= SEQUENCE { Abs—Coord, Coord, Coord }

—— This command carries the functionality of the Data Syntax I, and III ARC command (outline form) and of the Data Syntax II GDP (circular arc 3 point) command.

—— The absolute coordinate defines the initial position of the arc. The two other coordinate parameters define a point on the arc and the final position of the arc respectively.

A.3.10.1.3.3 *Arc—3 point chord*

Arc—3—Point—Chord ::= SEQUENCE { Abs—Coord, Coord, Coord }

—— This command carries the functionality of the Data Syntax I, and III ARC command (chord fill form) and of the Data Syntax II GDP (circular arc 3 point chord) command.

—— The absolute coordinate defines the initial position of the arc. The two other coordinate parameters define a point on the arc and the final position of the arc respectively. A Chord is drawn from the initial to the final position of the arc.

A.3.10.1.3.4 *Arc—3 point pie*

Arc—3—Point—Pie ::= SEQUENCE { Abs—Coord, Coord, Coord }

—— This command carries the functionality of the Data Syntax II GDP (circular arc 3 point pie) command.

—— The absolute coordinate defines the initial position of the arc. The two other coordinate parameters define a point on the arc and the final position of the arc respectively. Two lines are drawn from the initial to the geometric centre of the arc and then to the final position of the arc to form a pie shape. Although a pie filled arc is not directly available in data syntax DS I or DS III the conversion process can simulate the function by the use of an arc and two lines.

A.3.10.1.3.5 *Ellipse*

Ellipse ::= SEQUENCE { Abs—Coord, Coord, Coord, Coord }

—— This command carries the functionality of the Data Syntax II GDP (ellipse) command.

—— The absolute coordinate defines the initial position of the ellipse. A second coordinate parameters defines a point on the opposite side of the arc which establishes the major axis diameter. The third and fourth parameters define the minor axis diameter. Although an ellipse or elliptic arc are not directly available in data syntax DS I or DS III the conversion process can simulate the function in a piecewise manner or by fitting a spline curve.

A.3.10.1.3.6 *Elliptic arc*

Elliptic—Arc ::= SEQUENCE { Abs—Coord, Coord, Coord, Coord }

—— This command carries the functionality of the Data Syntax II GDP (elliptic arc) command.

—— The absolute coordinate defines the initial position of the arc. A second coordinate parameters defines a point on the opposite side of the arc which establishes the major axis diameter. A third parameter defines the minor axis diameter. A fourth parameter defines the final position of the arc.

A.3.10.1.3.7 *Elliptic arc chord*

Elliptic—Arc—Chord ::= SEQUENCE { Abs—Coord, Coord, Coord, Coord }

—— This command carries the functionality of the Data Syntax II GDP (elliptic arc chord) command.

—— The absolute coordinate defines the initial position of the arc. A second coordinate parameters defines a point on the opposite side of the arc which establishes the major axis diameter. A third parameter defines the minor axis diameter. A fourth parameter defines the final position of the arc. A Chord is drawn from the initial to the final position of the arc.

A.3.10.1.3.8 *Elliptic arc pie*

Elliptic—Arc—Pie ::= SEQUENCE { Abs—Coord, Coord, Coord, Coord }

—— This command carries the functionality of the Data Syntax II GDP (elliptic arc pie) command.

—— The absolute coordinate defines the initial position of the arc. A second coordinate parameters defines a point on the opposite side of the arc which establishes the major axis diameter. A third parameter defines the minor axis diameter. A fourth parameter defines the final position of the arc. Two lines are drawn from the initial to the geometric centre of the arc and then to the final position of the arc to form a pie shape.

A.3.10.1.3.9 *Arc centre cord*

Arc—Centre—Chord ::= SEQUENCE { Abs—Coord, Coord, Coord }

—— This command carries the functionality of the Data Syntax II GDP (arc—centre—chord) command.

—— The absolute coordinate defines the initial position of the arc. The other coordinate parameters define the start and end points of the arc.

A.3.10.1.3.10 *Arc centre pie*

Arc—Centre—Pie ::= SEQUENCE { Abs—Coord, Coord, Coord }

—— This command carries the functionality of the Data Syntax II GDP (arc—centre—pie) command.

—— The absolute coordinate defines the centre of the arc. The other coordinate parameters define the start and end points of the arc.

A.3.10.1.4 *Rectangle*

Rectangle ::= SEQUENCE { Abs—Coord, Rel—Coord }

—— This command carries the functionality of the Data Syntax I and III RECTANGLE command and the Data Syntax II GDP (rectangle) command.

—— The absolute coordinate defines the initial position of the rectangle. A relative coordinate parameters defines a point on the diagonally opposite side of the rectangle which establishes the size of the rectangle.

A.3.10.1.5 *Polygon*

Polygon ::= SEQUENCE { Abs—Coord, SEQUENCE OF { Rel—Coord } }

—— This command carries the functionality of the Data Syntax I and III POLYGON (filled) command and the Data Syntax II FILL AREA command. Data Syntax I and III also provide a POLYGON (outline) command which can be carried through the IDS by a LINE command with a repetition of the initial points as the final point.

—— The absolute coordinate defines the initial position of the polygon. The sequence of relative coordinate define the vertices of the polygon. A polygon is always closed and the final position is the same as the initial position.

A.3.10.1.6 *Spline*

Spline ::= SEQUENCE { Abs—Coord, SEQUENCE OF { Rel—Coord } }

—— This command carries the functionality of the Data Syntax I and III ARC (spline) command and the Data Syntax II GDP (spline) command.

—— The absolute coordinate defines the initial position of the poly curve. The sequence of relative coordinates (greater than 3) define the curve.

—— *Note* — The various terminal data syntaxes do not use exactly the same definition of the type and/or parameters for the spline generating function, however all of the source terminal data syntaxes tend to use a spline function of some type. Although potentially this could cause significant diferences in the resultant picture after conversion, it is still the closest result than can be generated in a reasonable manner.

A.3.10.1.7 *Pixel array*

```
Pixel—Array  ::= SEQUENCE {
                        first—point Abs—Coord,
                        second—point Abs—Coord,
                        third—point Rel—Coord,
```

—— these 3 points define the pixel area which in general could be a parallelogram. The first two points are the end points of a diagonal.

```
                        cells—first—direction INTEGER,
                        cells—second—direction INTEGER,
```

—— These values divide the pixel area in a grid with equal dimensions, to represent the intended (logical) resolution. The first direction is considered from the first to the third point. The second direction is from the first point to the unspecified point. These values can easily be derived, e.g. from the logical pel in case of INCREMENTAL POINT.

```
                        Pixel—Array—Data }

Pixel—Array—Data      ::= CHOICE { [1] IMPLICIT SEQUENCE OF Basic—Colour—Selection,
                                   [2] IMPLICIT SEQUENCE OF Direct—Colour—Selection,
                                   [3] IMPLICIT SEQUENCE OF Indexed—Colour—Selection }
```

—— The colour list is defined according to the `Colour—Control—String`. Auxiliary colour selection is not meaningful for this definition. The first colour is mapped to the cell associated with the first point. The colour elements are mapped within rows running from the first to the third point, and with rows incrementing in order from the third to the second point.

The various source terminal Videotex data syntaxes contain commands to efficiently code line and polygon data in an incremental fashion to achieve greater efficiency. The incremental capability differs greatly between the different data syntaxes, and no intermediate format could be developed which would be suitable in all the different environments. Since efficiency is of secondary importance, incremental lines and polygons should be communicated in terms of the general line and polygon functions above.

A.3.10.2 *Geometric control commands*

A large number of control commands are available in each of the terminal data syntaxes to control the geometric drawing functions. Although many of the geometric control commands defined in each of the data syntaxes may appear to be the same, they differ in side effect. For this reason all of the geometric control commands which appear in the various data syntaxes are included here. Only where the control commands are identical, such as a number of the geometric control commands in data syntaxes DS I and DS III, is a common control command definition used below.

```
Geometric—Control—Command  ::= CHOICE { [1] Geo—Control—Command—1,
                                         [2] Geo—Control—Command—2 }
```

—— Two types of geometric control commands are included in the IDS in order to accommodate the two different approaches taken in Data Syntax II and in Data Syntax I, III. These commands are grouped separately since they would never be received in combination.

A.3.10.2.1 *Geo control command—1*

```
Geometric—Control—Command—1 ::= CHOICE { [1] Numeric—Precision,
                                          [2] Drawing—Point—Size,
                                          [3] Line—Style,
                                          [4] Highlight,
                                          [5] Fill,
                                          [6] Field,
                                          [7] Blink—Process,
                                          [8] Wait }
```

—— Geometric control commands analogous to those in Data Syntax I and III.

A.3.10.2.1.1 *Numeric precision*

```
Numeric—Precision      ::= SEQUENCE { REAL, REAL }
```

—— This command carries the functionality of the Data Syntax I, and III DOMAIN geometric control command.

—— Define the nominal numeric precision in use by the source data syntax. Since the ASN.1 encoding rules permit any precision of data to be communicated, this control command does not affect the precision of data communicated. It is used to inform the conversion process of the nominal precision being used by the source data syntax. The first parameter carries the precision, expressed as a number of significant bits, for single—value operands. Similarly the second parameter carries the number of significant bits for multi—valued (2d and 3d) operands.

A.3.10.2.1.2 *Drawing point size*

```
Drawing—Point—Size  ::= Rel—Coord
```

—— This command carries the functionality of the Data Syntax I, and III DOMAIN (logical pel size) geometric control command.

—— This geometric control function establishes the size of the logical drawing point (LOGICAL PEL) as a fraction of the unit screen dimensions. The special case of zero is interpreted as being the smallest size possible on a given presentation device.

A.3.10.2.1.3 *Line style*

```
Line—Style      ::= INTEGER { solid          (1),
                              dotted         (2),
                              dashed         (3),
                              dot—dashed     (4) }
```

—— Establish the style for presenting lines from a fixed set of line styles.

—— This command carries the functionality of the Data Syntax I, and III TEXTURE (line texture) geometric control command.

A.3.10.2.1.4 *Highlight*

```
Highlight       ::= BOOLEAN
```

—— Establish whether filled areas are drawn in highlight mode, in which the perimeter is drawn in BLACK or a contrasting colour to the fill.

—— This command carries the functionality of the Data Syntax I, and III TEXTURE (highlight) geometric control command.

A.3.10.2.1.5 *Fill*

```
Fill            ::= BOOLEAN
```

—— Establish whether polygons, closed arcs, ellipses or rectangles are to be filled. For efficiency this control is codes as part of the opcode identifying the drawing primitive in some of the source terminal data syntaxes. This function has been separated here in order to ease conversion between data syntaxes.

—— This command carries the functionality of the Data Syntax I, and III TEXTURE (fill texture pattern) geometric control command.

A.3.10.2.1.6 *Field*

```
Field           ::= Rel—Coord
```

—— Define the dimensions of the active area on the dispaly screen. The field command establishes boundaries which “contain” text; that is boundaries for scroll areas, and to which the format effector characters operate. The initial position is defined by the current geometric drawing position. The relative coordinate parameters define a point on the diagonally opposite side of the field which establishes the size of the field rectangular area.

—— This command carries the functionality of the Data Syntax I, and III FIELD geometric control command.

A.3.10.2.1.7 *Blink process*

```
Blink—Process ::= SEQUENCE { [1] INTEGER,
                             [2] INTEGER OPTIONAL,
                             [3] INTEGER OPTIONAL,
                             [4] INTEGER OPTIONAL }
```

—— Establish a Blink process in which the colour map is dynamically altered for a specified interval and phase. The first integer represents the colour map address of the Blink To colour, then the On interval, the Off interval and the Phase Delay in 1/10 of a second respectively. The capability to handle Blink processes is very terminal model dependent. In general Blink processes can be used to simulate any other blink capability available in any data syntax, within the limits of the available memory assigned for such operations, as specified in boundary value conditions. However Blink processes cannot easily be simulated in display environments which do not present sufficient capabilities.

—— This command carries the functionality of the Data Syntax I, and III BLINK geometric control command.

A.3.10.2.1.8 *Wait*

```
Wait        ::= INTEGER
```

—— Establish a time delay in processing presentation data for the time specified in units of 1/10 of a second. Although the Wait command is very simple, it provides very great problems in conversion. This is because the wait command is a dynamic control command. Presentation dynamics cannot be guaranteed in conversion because the order of presentation commands may have to be altered to accommodate for differences in the terminal model between two data syntaxes. Conversion of the wait command should only be attempted when the source and target presentation processes are in synchronization, i.e. when no sorting of presentation commands is necessary in the conversion, or at the end of a unit (page) of data.

A.3.10.2.2 *Geo control command—2*

```
Geo—Control—Command—2 ::= CHOICE { [1] Display—Element—Attributes,
                                    [2] Control—Element—Attributes }
```

—— Geometric control commands analogous to those in Data Syntax II.

—— Display Element Attributes pertain to the output display primitives. Some of this primitives may be similar to those in Geo—Control—Command—1 section, however the side effects are different for these commands.

—— Control Element Attributes establish the display transformation, clipping and work station control functions which are unique to the display environment associated with Data Syntax II.

—— The use of bundle facilities is for further study.

A.3.10.2.2.1 *Display element attributes*

```
Display—Element—Attributes      ::= CHOICE {
                                        [1] IMPLICIT Line—Attributes,
                                        [2] IMPLICIT Marker—Attributes,
                                        [3] IMPLICIT Fill—Area—Attributes }

Line—Attributes                 ::= SET {
                                        [1] IMPLICIT Line—Type OPTIONAL,
                                        [2] IMPLICIT Line—Width—Scale—Factor OPTIONAL,
                                        [3] IMPLICIT Polyline—Colour—Index OPTIONAL }

Line—Type                       ::= INTEGER {
                                        solid                          (0),
                                        dashed                         (1),
                                        dotted                         (2),
                                        dashed—dotted                  (3),
                                        implementation dependent       (4) }

Line—Width—Scale—Factor         ::= REAL

Polyline—Colour—Index           ::= Colour—Index

Marker—Attributes                   ::= SET {
                                        [1] IMPLICIT Marker—Type OPTIONAL,
```

```
                                    [2] IMPLICIT Marker—Size—Scale—Factor OPTIONAL,
                                    [3] IMPLICIT Polymarker—Colour—Index OPTIONAL }

Marker—Type                         ::= INTEGER {
                                        dot                                     (0),
                                        plus                                    (1),
                                        asterisk                                (2),
                                        circle                                  (3),
                                        diagonal—cross                          (4) }

Marker—Size—Scale—Factor    ::= REAL

Polymarker—Colour—Index     ::= Colour—Index

Fill—Area—Attributes        ::= SET {
                                        [1] IMPLICIT Fill—Area—Interior—Style OPTIONAL,
                                        [2] IMPLICIT Fill—Area—Colour—Style OPTIONAL,
                                        [3] IMPLICIT Fill—Area—Style—Index OPTIONAL,
                                        [4] IMPLICIT Pattern—Reference—Point OPTIONAL,
                                        [5] IMPLICIT Pattern—Vectors OPTIONAL }

Fill—Area—Interior—Style    ::= INTEGER {
                                        hollow                                  (0),
                                        solid                                   (1),
                                        pattern                                 (2),
                                        hatch                                   (3) }

Fill—Area—Colour—Index      ::= Colour—Index

Fill—Area—Style—Index               ::= INTEGER {
```

—— For interior style pattern the fill area style index selects a pattern defined by “Fill pattern control string”.

—— For interior style hatch the following styles are selected:

```
                                        vertical—lines                          (0),
                                        horizontal—lines                        (1),
                                        slope—45—degree—lines                   (2),
                                        slope—45—degree—lines                   (3),
                                        crossed—lines—vertical—and—horizontal—lines     (4),
                                        crossed—lines—45—and—45—degrees         (5) }

Pattern—Reference—Point             ::= Abs—Coord

Pattern—Vectors             ::= SEQUENCE { Abs—Coord, Abs—Coord }
```

—— The origin of the NDC space an the first point defines the pattern height vector. The origin of the NDC space an the second point defines the pattern widht vector.

```
Colour—Index                        ::= CHOICE {
                                        [1] IMPLICIT Basic—Colour—Selection,
                                        [2] IMPLICIT Indexed—Colour—Selection }
```

A.3.10.2.2.2 *Control element attributes*

```
Control—Element—Attributes  ::= CHOICE {
                                        [1] WS—Management—Primitives,
                                        [2] Transformation—Primitives }

WS—Management—Primitives    ::= CHOICE {
open—workstation                        [1] IMPLICIT INTEGER,
                                         —— WS Identifier
close—workstation                       [2] IMPLICIT INTEGER,
                                         —— WS Identifier
activate—workstation                    [3] IMPLICIT INTEGER,
                                         —— WS Identifier
deactivate—workstation                      [4] IMPLICIT INTEGER,
                                         —— WS Identifier
clear—workstation                       [5] IMPLICIT INTEGER,
```

```
                                        —— WS Identifier
set—defaults                            [6] IMPLICIT NULL,
update—workstation                      [7] IMPLICIT Update—WS,
deferral—state                          [8] IMPLICIT Deferral—State }

Update—WS               ::= SEQUENCE {
workstation—identifier                  INTEGER,
regeneration—flag                       INTEGER { perform       (0),
                                                  postpone      (1) } }

Deferral—State          ::= SEQUENCE {
workstation—identifier                  INTEGER,
deferral—mode                           INTEGER { asap          (0),
                                                  bnil          (1),
                                                  bnig          (2),
                                                  asti          (3) }
implicit—regeneration                   INTEGER { suppressed    (0),
                                                  allowed       (1) } }

Transformation—Primitives   ::= SET {
                                        [1] IMPLICIT WS—Window OPTIONAL,
                                        [2] IMPLICIT WS—Viewport OPTIONAL,
                                        [3] IMPLICIT Clipping—Rectangle OPTIONAL }

WS—Window                       ::= SEQUENCE {
workstation—Identifier                  INTEGER,
first—point                             Abs—Coord,
second—point                            Abs—Coord }

WS—Viewport             ::= SEQUENCE {
workstation—identifier                  INTEGER,
xmin                                    REAL,
xmax                                    REAL,
ymin                                    REAL,
ymax                                    REAL }

Clipping—Rectangle      ::= SEQUENCE {
```

```
first—point                    Abs—Coord,
second—point                   Abs—Coord }
```

A.3.10.3 *Geometric coordinates*

Coordinate data for geometric operations is stored in terms of normalized display coordinates in all three source data syntaxes. However, the exact details of the number format differ significantly between the approach taken in data syntaxes DS I and DS III and that taken in data syntax DS II. Since the purpose of the IDS is for interworking, differences with respect to the number format should be avoided. Therefore, within the IDS a simple numbering scheme based on the ASN.1 signed REAL data type is used. ASN.1 REAL numbers are self—delimiting and of arbitrary length, so there is no difficulty with precision and no need to assign special bit fields to determine the length of the number. A coordinate can therefore be represented as a pair of numbers. The mapping of a real data field to a numeric data field in any of the data syntaxes is dependent upon that particular data syntax. For the case of data syntaxes DS I and DS III the normalized unit display area is mapped to the fractional part (i.e. mantissa part) of the real number field. For DS II both the mantissa and exponent of the real number are used.

Since three dimensional coordinate specifications are optionally available in all of the data syntaxes, an integer triplet is optionally provided below. Because three dimensional operation is optional, the projection to two dimensions must be defined so that three dimensional information may be viewed in a two dimensional environment through interworking. A plane projection which assumes Z = 0 is used.

```
Coord          ::= IMPLICIT CHOICE { Abs—Coord, Rel—Coord }

Abs—Coord      ::= CHOICE { [1] X—Y,
Abs—Coord                   [2] X—Y—Z }

X—Y            ::= SEQUENCE { REAL, REAL }

——      Absolute X, Y Coordinates

X—Y—Z          ::= SEQUENCE { REAL, REAL, REAL }

——      Absolute X, Y, Z Coordinates

Rel—Coord      ::= CHOICE {

                     [3] DX—DY,
                     [4] DX—DY—DZ }

DX—DY          ::= SEQUENCE { REAL, REAL }

——      Relative DX, DY Coordinates

DX—DY—DZ       ::= SEQUENCE { REAL, REAL, REAL }

——      Relative DX, DY, DZ Coordinates
```

A.3.11 *Animation control string*

The capability to achieve dynamic or animated effects on the presentation device is highly dependent on the terminal model and display environment. Several of the terminal data syntaxes provide some specialized capabilities to achieve dynamic effects. For example, data syntaxes DS I and DS III include three phase flash (blink) capability and data syntax DS III includes a colour map phased blink function. The dynamic effects generated by these special functions will not in general be preserved in conversion. This is especially true since the order of the display of presentation entities may be altered by the conversion process to account for differences in the terminal model. Except for flash (blink) it is necessary for the conversion process to take into account dynamic effects even though it cannot convert them faithfully, since they may significantly alter the final resultant picture.

A sophisticated terminal model dependent animation capability is available in data syntax DS I. This capability makes use of a multi—plane terminal model in which the order and relative position of the various planes may be altered. The effects which may be generated by this capability are unique to the environment in which they were defined. The animation control commands from data syntax DS I must, however, be included in the interworking data syntax, since they affect the final result of the display. The conversion process must generate the correct final resultant picture.

```
Animation—Control—String   ::= CHOICE { mvi—start          [1] NULL,
                                         mvi—stop           [2] NULL,
                                         mvi—repeat—start   [3] MVI—Repeat—Start,
                                         mvi—repeat—end     [4] NULL,
```

mvi—move [5] MVI—Move }

—— MVI—Start is a function from Recommendation T.101 DS I (MVI Code set position 2/0)

—— MVI—Stop is a function from Recommendation T.101 DS I (MVI Code set position 2/1)

A.3.11.1 *MVI—repeat start*

MVI—Repeat—Start ::= SEQUENCE { GRAPHICSTRING, INTEGER }

—— General character (REPEAT START) from Recommendation T.101 DS I (MVI Code set position 3/12 or 11/12), followed by a count of the number of repetitions

—— MVI—Repeat—End is a function from Recommendation T.101 DS I (MVI Code set position 3/13 or 11/13)

A.3.11.2 *MVI—move*

MVI—Move ::= SEQUENCE { Move—Origin, Move—Termination, Move—Time }

—— MVI—Move is a function from Recommendation T.101 DS I (MVI Code set position 3/10 or 11/10)

Move—Origin ::= Abs—Coord

—— X, Y Parameters codes as packed binary fractions

Move—Termination ::= OCTETSTRING

—— X, Y Parameters codes as packed binary fractions

Move—Time ::= INTEGER

—— Numeric count of the time period for the move operation in units of 1/10 of a second

A.3.12 *Segment control string*

Data syntax II provides an optional segment storage and editing capability. One or two storage memories for display segments are retained. Editing commands may produce dynamic effects by altering the stored display segment and causing the redisplay of the picture. A display segment may contain any geometric string data as well as the special segment attributes as described below.

Segment control is similar to animation control in that it provides functions which control special display environment dependent capabilities. Since analogous functions are not available in either data syntax I or III, these functions must be handled in the conversion process. For the conversion of information from data syntax II into data syntax I or III only one “Workstation” (or display screen) is used.

```
Segment—Control—String          ::= CHOICE { [1] Work—Station—Dependent,
                                             [2] Work—Station—Independent }

Work—Station—Dependent          ::= CHOICE { [1] W—Create,
                                             [2] W—Close,
                                             [3] W—Rename,
                                             [4] W—Delete—1,
                                             [5] W—Delete—2,
                                             [6] W—Redraw,
                                             [7] W—Set—Highlight,
                                             [8] W—Set—Visibility,
                                             [9] W—Set—Seg—Transparent,
                                             [10] W—Set—Priority }
```

A.3.12.1.1 *W—create*

W—Create ::= INTEGER

—— Open the identified segment.

A.3.12.1.2 *W—close*

W—Close ::= INTEGER

—— Close the identified segment.

A.3.12.1.3 *W—rename*

```
W—Rename  ::= SEQUENCE {
                    old—segment—number      [1] INTEGER,
                    new—segment—number      [2] INTEGER }
```

—— Rename old segment number to new segment number.

A.3.12.1.4 *W—delete—1*

```
W—Delete—1          ::= SEQUENCE {
                    work—station—id         [1] INTEGER,
                    segment—number          [2] INTEGER }
```

—— Delete identified segment from workstation.

A.3.12.1.5 *W—delete—2*

```
W—Delete—2          ::= INTEGER
```

—— Delete the identified segment from all workstations.

A.3.12.1.6 *W—redraw*

```
W—Redraw        ::= INTEGER
```

—— Redraw the identified workstation.

A.3.12.1.7 *W—set highlight*

```
W—Set—Highlight     ::= SEQUENCE {
                        highlight—segment—number    [1] INTEGER,
                        highlight—attribute         [2] INTEGER }
```

—— Set highlight attribute of identified segment.

A.3.12.1.8 *W—set visibility*

```
W—Set—Visibility    ::= SEQUENCE {
                        visibility—segment—number   [1] INTEGER,
                        lity—attribute              [2] INTEGER }
```

—— Set visibility attribute of identified segment.