INTERNATIONAL STANDARD

ISO 11783-6

Fourth edition
2018-06

Tractors and machinery for agriculture and forestry — Serial control and communications data network —

Part 6:
Virtual terminal

Tracteurs et matériels agricoles et forestiers — Réseaux de commande et de communication de données en série —

Partie 6: Terminal virtuel

Reference number
ISO 11783-6:2018(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 23, *Tractors and machinery for agriculture and forestry*, Subcommittee SC 19, *Agricultural electronics*.

This fourth edition cancels and replaces the third edition (ISO 11783-6:2014) which has been technically revised. New requirements in this fourth edition are specified as VT version 6. Changes include clarifications to existing VT object and command behaviour, including additional capabilities. One Working Set in collaboration with another Working Set can transfer control of the active mask to the other with the Select Active Working Set command. Additional objects include a Colour Palette object, a Graphic Data object, a Scaled Graphic object, and a Working Set Special Controls object.

A list of all the parts in the ISO 11783 series can be found on the ISO website.

Introduction

ISO 11783-1 to ISO 11783-14 specify a communications system for agricultural equipment based on the ISO 11898[5] protocol. SAE J1939[1] documents, on which parts of ISO 11783 are based, were developed jointly for use in truck and bus applications and for construction and agriculture applications. Joint documents were completed to allow electronic units that meet the truck and bus SAE J1939 specifications to be used by agricultural and forestry equipment with minimal changes. The specifications for virtual terminals given in this part of ISO 11783 are based on DIN 9684-4[2]. General information on ISO 11783 is to be found in ISO 11783-1.

The purpose of ISO 11783 is to provide an open, interconnected system for on-board electronic systems. It is intended to enable electronic control units (ECUs) to communicate with each other, providing a standardized system.

All phrases in this document that refer explicitly to a software term for an object or a command have the first letter of each object or command word capitalized (e.g. Output Linear Bar Graph object, Change Numeric Value command). This aides in the recognition of these terms as a specific item which has a specific definition in this document.

Tractors and machinery for agriculture and forestry — Serial control and communications data network —

Part 6: Virtual terminal

1 Scope

ISO 11783 as a whole specifies a serial data network for control and communications on forestry or agricultural tractors, mounted, semi-mounted, towed or self propelled implements. Its purpose is to standardize the method and format of transfer of data between sensor, actuators, control elements, information storage and display units whether mounted or part of the tractor, or any implements.

This document describes a universal virtual terminal that can be used by both tractors and implements.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 11783-3, *Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 3: Data link layer*

ISO 11783-5, *Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 5: Network management*

ISO 11783-7, *Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 7: Implement messages application layer*

ISO 15077, *Tractors and self-propelled machinery for agriculture — Operator controls — Actuating forces, displacement, location and method of operation*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
auxiliary input unit
autonomous control function (CF) providing Auxiliary Controls for common use that can also be physically located within an electronic control unit (ECU), or on the virtual terminal (VT)

3.2
object pool
collection of objects that completely define the operator interface for an implement or a single Working Set

Note 1 to entry: The complete VT definition will be made up of one or more object pools — one for each Working Set.

3.3
Object ID
numeric value which identifies a specific object within an object pool

Note 1 to entry: Object ID values range from 0 to $FFFF_{16}$ (65535_{10}), with 65535 as the NULL Object ID.

3.4
attribute ID
AID
numeric value which references a specific object's attribute

Note 1 to entry: AID values range from 0 to FF_{16} (255_{10}), with 255 as the NULL_AID.

3.5
char
single character where the size is 1 byte

Note 1 to entry: Commonly used for ISO 8859 characters (e.g. 41_{16} in ISO 8859-1 represents “A”) (see Annex K).

3.6
character
single text grapheme or symbol, as in an alphabet

Note 1 to entry: Size is variable based on the encoding scheme [see *char* (3.5) and *WideChar* (3.11)].

3.7
code plane
group of 65536 possible character codes

Note 1 to entry: Unicode/ISO 10464 organizes the characters in 17 code planes numbered 0 to 16.

EXAMPLE Code plane 0 covers characters 000000_{16} to $00FFFF_{16}$

Code plane 1 covers characters 010000_{16} to $01FFFF_{16}$

...

Code plane 16 covers characters 100000_{16} to $10FFFF_{16}$.

3.8
open input object
state of an input object where the object has focus and it is open for operator input

Note 1 to entry: Open input object is used interchangeably with data input.

3.9
selected input object
state of an input object where the object has focus but it is not open for operator input

Note 1 to entry: Selected input object is used interchangeably with “has focus”.

3.10
surrogate pair
32 bit code for characters composed of a 16 bit high pair and a 16 bit low pair

Note 1 to entry: UTF-16 encoding of characters in code plane 1 to 16 (see 4.6.19.7).

Note 2 to entry: UTF-16 Character encoding scheme defined by ISO 10646.

3.11
WideChar
single character with a size of 2 bytes encoded in little endian order

EXAMPLE Byte sequence 41_{16}, 00_{16} represents “A” (see Annex K).

Note 1 to entry: Two WideChars can be combined to indicate character codes exceeding 16-bit (see 4.6.19.7).

3.12
WideString
zero or more characters composed of the primitive type “WideChar” always preceded by the byte order mark $FEFF_{16}$

EXAMPLE Byte sequence FF_{16},FE_{16},41_{16},00_{16},42_{16},00_{16},43_{16},00_{16} represents “ABC”. This WideString has a Length of 8 bytes with the number of characters in the presentation equal to 3.

3.13
8-bit string
zero or more characters composed of the primitive type “char”

Note 1 to entry: String length is variable.

3.14
VT Number
number that is used to uniquely identify each connected VT to the operator

Note 1 to entry: See 4.6.25 and D.18.

3.15
User-Layout Data Mask
special Data Masks that are controlled by the VT but laid out by the operator

Note 1 to entry: See 4.1 for information on data mask variations, and 4.7 for User-Layout Data Mask information.

3.16
Window Cell
rectangular presentation cell in a grid on a User-Layout Data Mask

Note 1 to entry: See 4.7.

3.17
Window Mask object
rectangular presentation area composed of one or more adjacent Window Cells

Note 1 to entry: See 4.7.

3.18
User-Layout Soft Key Mask
Soft Key Masks that are controlled by the VT but laid out by the operator

Note 1 to entry: See 4.7.

3.19
Key Cell
cell that is the size of a Soft Key designator in a User-Layout Key Mask

Note 1 to entry: See 4.7.

3.20
Key Group object
area of one or more Key Cells and contains a grouping of one or more Key objects

Note 1 to entry: See 4.7.

3.21
Non-VT Screen
display screen that is not part of the VT application or one in which the layout is controlled by the VT

EXAMPLE A screen that comes from another application within the display (see 4.7).

3.22
Non-VT Area
visible area outside the normal Data Mask and Soft Key Mask areas

EXAMPLE A display of information related to the vehicle operation (see 4.7).

3.23
referenced WS
working set with an object pool containing objects which are shown by another object pool via the External Object Pointer object

Note 1 to entry: See 4.6.11.6.

3.24
referencing WS
working set with an object pool which shows object(s) from another object pool via the External Object Pointer object

Note 1 to entry: See 4.6.11.6.

3.25
Functionally Identical WS
Working Set(s) with a NAME that exactly matches other Working Sets, when the Self Configurable, Instance fields, and Identity Number are excluded in the comparison

3.26
Line End
"cursor" or text positioning control intended to locate the following displayable character "font height" pixels downward and at the left-most position in the containing object

Note 1 to entry: See 4.6.19.6.

3.27
Model Identification Code
proprietary code defined by the manufacturer that defines a unique model and version of an Auxiliary Input Unit that does not change at runtime, and is revised by the manufacturer when a new and incompatible Auxiliary Input Unit is created

4 Technical requirements

4.1 Overview

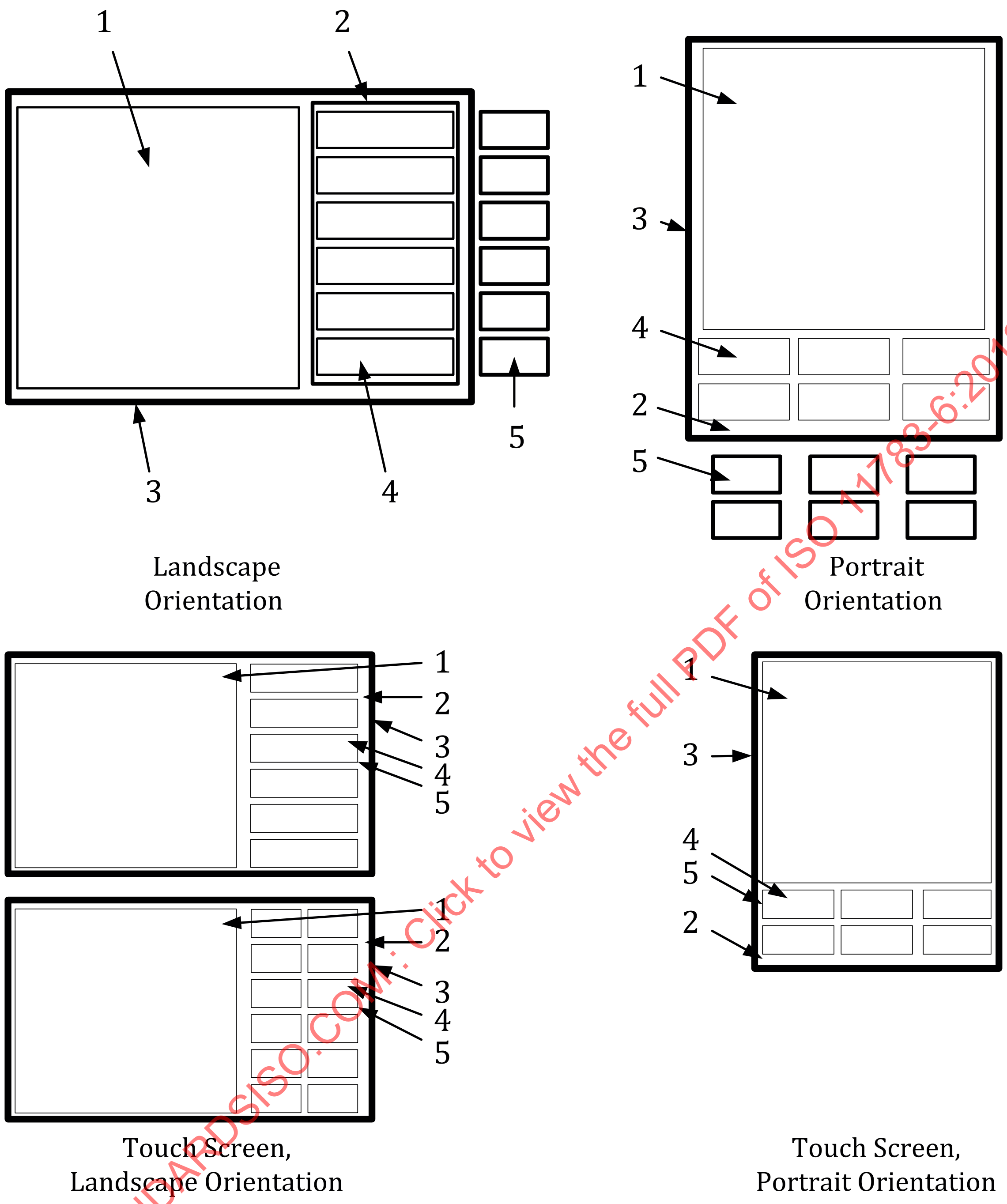
A virtual terminal (VT) is a control function (CF) within an electronic control unit (ECU), consisting of a graphical display and input functions, connected to an ISO 11783 network that provides the capability for a CF, composing an implement or a group of implements to interact with an operator. The VT provides the capability to display information and to retrieve data from an operator. The CF, as an implement or a group of implements represented by a Working Set Master acquires storage for objects within the VT and on demand displays this stored information to an operator. In this document, the term *Working Set* will be used for a CF, as an implement or a group of implements either represented by a single ECU or a group of ECUs acting as a Working Set. Working Sets on the network can also acquire the use of input methods of the VT to allow the operator to send signals back to the Working Set.

This document describes the VT with the detail and clarity required for VTs built by different manufacturers to be interchangeable with any implement Working Set that uses its services. The interface protocol of this document also reduces the run-time ISO 11783 communication bus traffic as much as possible. For these reasons, the requirements of this document are organized in an object-oriented manner with specific attributes and behaviour of each object clearly and fully defined. The required behaviour of the VT given certain situations is also detailed.

In general, the functions, not the design, of the user interface of the VT are defined in order to avoid restrictions on possible designs. However, certain limitations are imposed in order to meet the goal of interchangeability between various manufacturers. Specifications regarding physical layout, components, processing power and the number of physical elements comprising a VT have been omitted in order to avoid restricting manufacturer's designs.

The VT shall have a pixel-addressable (graphical) display. Information from connected Working Sets is shown to the operator on the graphical display. This information is shown in display areas that are defined by Data Masks, Alarm Masks and Soft Key Masks. The data for these masks is contained in object definitions that are loaded into a VT via the ISO 11783 CAN bus, or from non-volatile memory. When the information defined by a mask is required on the display, the mask can be made visible by a single Change Active Mask command from the Working Set, and therefore does not require significant additional network traffic.

The physical size, resolution, orientation and methods of implementing the graphical display are at the discretion of the designer of the VT. Figure 1 shows examples of some possible VT designs and orientations.


1
2
3
4
5
Landscape Orientation
1
3
4
2
5
Portrait Orientation
1
2
3
4
5
1
2
3
4
5
Touch Screen, Landscape Orientation
1
3
4
5
2
Touch Screen, Portrait Orientation

Key

1	data mask area	4	soft key designator
2	soft key mask area	5	physical soft key
3	physical screen		

Figure 1 — Virtual terminal — Examples

4.1.1 Technical requirements of VT versions

Essential characteristics and attributes of a VT can be identified using the Technical data messages as defined in Annex D.

VT version 6 imposed requirements are as follows.

Feature	Detailed requirements and recommendations	Clause(s)
Physical Soft Keys	Minimum 60 × 60 pixels (recommended to be square)	D.5
Text Fonts	All small font sizes, all large font sizes, all font styles.	D.7
Graphic Type	Minimum 256 colours, recommended 16-bit or greater colour capability to support Colour Palette object and Graphic Data object. The VT can downscale the presentation to match the hardware capability (e.g. 24-bit PNG downscaled to 16-bit VT hardware).	D.9
Data Mask Size	Minimum 480 × 480 pixels	4.5.2, D.9
Window Mask object	Fully parsed, presentation is optional.	4.7.3
Key Group object	Fully parsed, presentation is optional.	4.7.9
Graphics Context object	Full presentation support is required	B.18

4.2 Operator input and control

The VT shall provide the operator with means for control and input. There are five means associated with a VT that can be used for the input of data, selection of display data, and the control of connected Working Sets.

See Figure 2.

a) Soft — is a means, most likely keys on the VT, using software-changeable designators (labels). "Soft Keys" have their identity changed depending on which Soft Key Mask is visible. The VT shall make the association between a Soft Key and its designator clearly evident to the operator.

b) Navigation — is a means of selecting an input field or Button within the active Data Mask. If keys are used for "Navigation", they do not send key activation information to the Working Set and are proprietary to the VT.

c) Data Input — is a means of entering/editing information in an input field within the active Data Mask. If keys are used for "Data Input", they do not send key activation information to the Working Set and are proprietary to the VT. A means shall be provided for entering any number or character sequence that is valid for the input field.

During the data input operation, the VT Status message will continue to indicate the active Working Set, and active mask which contains the input object for which the data input operation applies.

There are two types of Data Input — "editing" and "real time editing".

1) **Editing** — is a means of data input where the new value being entered is composed by the operator using a proprietary means within the VT. During the composition of the new value, changes to the original value are not communicated to the Working Set. A means shall also be provided for ESC from or ENTER of information into a data field.

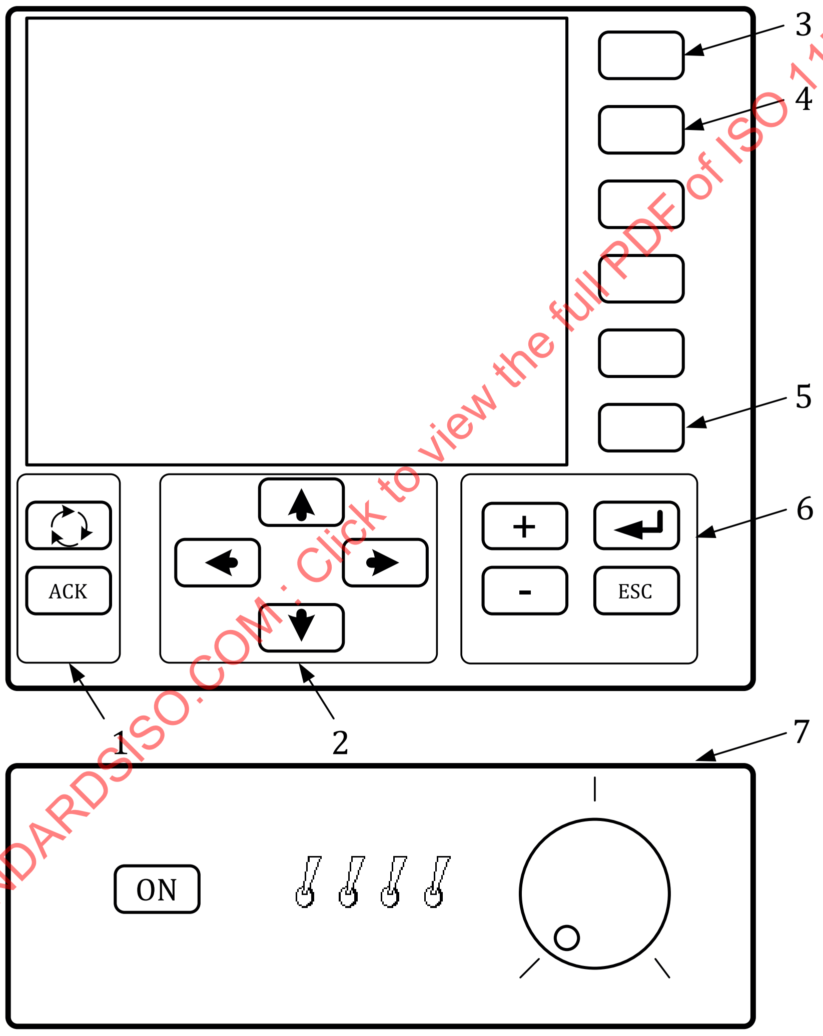
The ENTER means shall be provided to indicate to the Working Set the completion of data entry and communication of the new value, and the ESC means shall be provided to indicate that the data entry was aborted. The ENTER and ESC means can either be a permanent key or can only be available during data entry (see Table 5) The VT shall send a VT ESC message to a Working Set for an operator-activated ESC means or an ESC response as a response to receiving an ESC command from a Working Set.

2) **Real-time editing** — is a means of data input for an Input Number object and Input List object where the object has focus and it is open for operator input and changes by the operator to the value are periodically transmitted to the Working Set while the object is being changed. The VT Change Numeric Value message is limited to a 5 Hz update rate. Each value change sent to the Working Set is considered a complete transaction, as if the ENTER means was activated, and cannot be reverted by the ESC means. The VT is not required to provide steps in uniform

increments, however it shall be possible to set any value (e.g. fast scrolling is allowed to span a wide range of values, with fine adjustment for final setting). If the ESC means is activated during real time editing, the VT shall ensure that the on-screen value is equal to the value last sent to the Working Set. The VT can send a final value to the Working Set prior to sending the VT ESC message, or ESC response message to ensure this synchronization. Real time editing shall meet the operator controls requirement specified in ISO 15077.

d) Control — is a means of selecting between Working Sets whenever a Data Mask is available and for acknowledging alarms. Both means are required. Since more than one Working Set can use the services of the VT, the VT shall provide a means for the operator of selecting between connected Working Sets. The Working Set selection means should be indicated by three circular arrows or a similar graphic. Only the ACK means sends key activation information to the Working Set.

e) Auxiliary Input — is a means available to the operator for communicating input commands to the Working Set(s) using Auxiliary Controls which are assigned to Auxiliary Functions (see Annex J).



Key

1	control	5	soft key 6
2	navigation	6	data input
3	soft key 1	7	auxiliary input
4	soft key 2		

Figure 2 — Operator input and control means — Example

4.3 Acoustic alarm

The VT shall provide an acoustic alarm. The alarm can be a simple on/off type buzzer or an acoustic component capable of either/or variable frequency and audio level (see D.9).

4.4 Coordinate system

Positions and sizes in this document are always given in physical pixels unless otherwise stated. A two-dimensional coordinate plane (x, y) is used, where x is the number of units wide (x increases from left to right) and y is the number of units high (y increases from top to bottom). The coordinates are signed values. The origin (0, 0) for any object's coordinate system is located at the top left-hand corner of the parent object.

4.5 Display areas

4.5.1 General

This section defines standard Data Mask and Soft Key Mask areas of the display. Alternate usage of this area supports displaying data from multiple working sets (see 4.7).

4.5.2 Data Mask

The VT shall reserve an area of the display for displaying Data Masks and Alarm Masks. This area is called the Data Mask area (see Figure 1). Recognizing that the physical orientation of the VT display could be different, depending on the manufacturer of the VT, a square data mask aspect ratio is chosen to ensure correct display in either landscape or portrait orientation. The minimum Data Mask area shall be 200 pixels × 200 pixels (480 × 480 for VT version 6 and later). This requirement does not limit the physical resolution or size of the display, only the useable Data Mask area. Higher resolution mask areas are permitted, but the square aspect ratio shall be strictly enforced.

Examples of Data Mask areas that would meet this requirement are:

— 200 × 200;

— 240 × 240;

— 320 × 320;

— 480 × 480;

— 600 × 600;

— 800 × 800;

— 1 024 × 1 024.

Any other square dimensions would be acceptable.

It is suggested that unused areas of the physical display be used for proprietary information such as vehicle data, VT statistics or other data.

4.5.3 Soft Key Mask area and Soft Key designators

4.5.3.1 Soft Key variants and navigation

The VT shall reserve an area of the display for Soft Key labels, separate from the Data Mask area. This area is called the Soft Key Mask area (see Figure 1). Each Soft Key shall have a reserved display area, called a Soft Key designator, for displaying a label (see Figure 1). The minimum size of the designator field is 60 pixels wide × 32 pixels high regardless of screen orientation (60 × 60 for VT version 6 and later). The Soft Key designators can contain text, graphics or both. The Soft Key Mask area can be adjacent to, or physically separate from, the Data Mask area, but shall not be part of the Data Mask area.

The VT shall provide a clearly visible separation between the individual Soft Key designators (for example by drawing a one-pixel line). This visible separation shall be drawn outside of the Soft Key designator area. Only if the minimum size of the designator field cannot be fulfilled due to this requirement, the drawing of a one-pixel line on the border of the Soft Key designator area is acceptable.

The presentation of the Soft Keys can be further described in three groups, with a defined relationship: Navigation Soft Keys < Number of Physical Soft Keys ≤ Number of Virtual Soft Keys.

a) VT Version 3 and prior VTs have no requirement on the number of physical Soft Keys.

b) VT Version 4 and later VTs shall provide at least 6 physical Soft Keys.

c) VT Version 3 and prior shall support a maximum of 64 virtual Soft Keys per Soft Key Mask and shall support as a minimum the number of reported physical Soft Keys (see 4.5.3.3).

d) VT Version 4 and later shall support exactly 64 virtual Soft Keys per Soft Key Mask (see 4.5.3.3).

e) The VT shall provide a means for the operator to navigate and select all defined Soft Keys. For example, if there are six physical keys, some type of paging would be required to allow the operator to navigate to, and select from, any of the 64 virtual Soft Keys using the six physical keys.

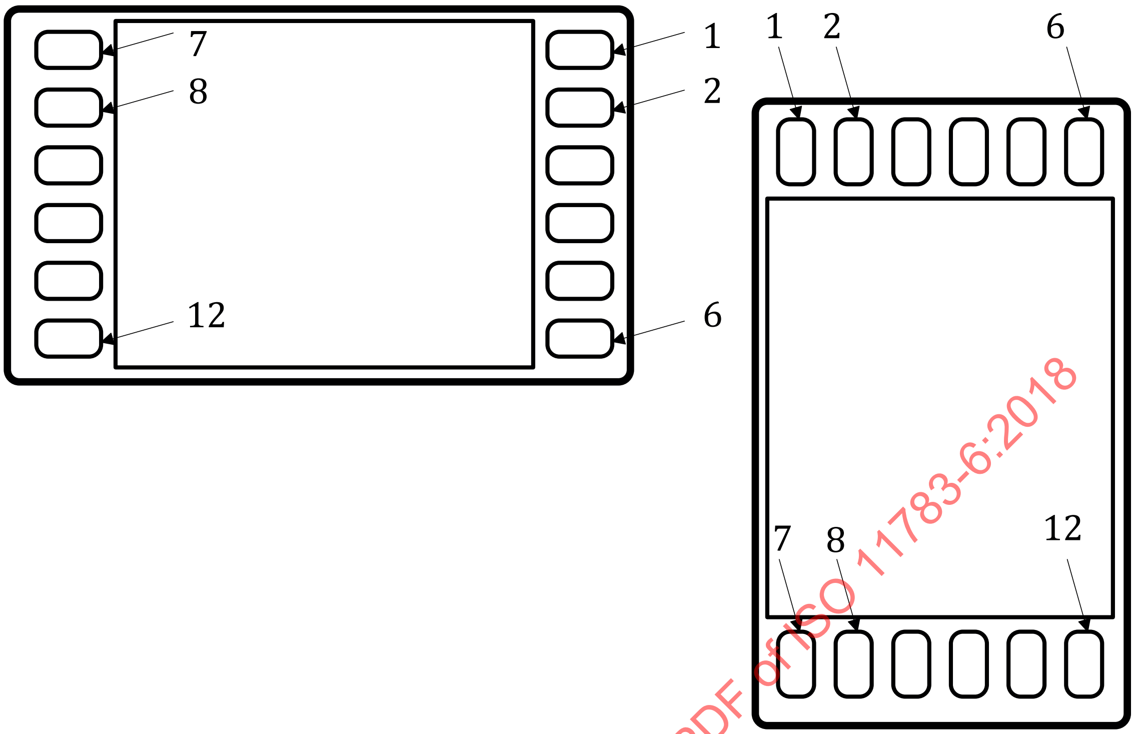
4.5.3.2 Physical Soft Keys

Physical Soft Keys is the count of the number of permanently dedicated keys that the VT makes available to active Working Sets. The term “physical Soft Key” does not imply that the VT necessarily provides physical buttons for the Soft Keys. For example on a VT with touch screen, the physical Soft Keys can be located directly on the touch screen as shown in Figure 1.

For VTs with a vertical arrangement of physical Soft Keys, key number 1 shall be on the right and the top-most position. Key number 2 shall be adjacent and below Key 1. Key m shall be at the bottom of the first column. If there are additional physical Soft Keys, the column containing keys m+1 to key n shall be to the left of the first column. Each additional column of physical Soft Keys shall continue to the left.

For VTs with a horizontal arrangement of physical Soft Keys, Key number 1 shall be on the top row and in the left-most position. Key number 2 shall be adjacent and to the right of Key 1. Key m shall be at the far right of the top row. If there are additional physical Soft Keys, the row containing keys m+1 to key n shall be below the first row. Each additional row of physical Soft Keys shall continue below the previous row. Examples of these arrangements are shown in Figure 3.

For VTs without a clear horizontal or vertical arrangement of physical Soft Keys (e.g. physical Soft Keys located in a matrix on the touch screen) the rules for a VT with a vertical arrangement of physical Soft Keys apply.



Key

1	key 1 physical location based on soft key arrangement
2	key 2 physical location based on soft key arrangement
6–8	key locations based on soft key arrangement
12	

Figure 3 — Physical Soft Key Orientation Examples showing Key Locations

4.5.3.3 Virtual Soft Keys

Virtual Soft Keys is the count of the number of Soft Keys that the VT supports for each active Working Set's Data Mask. If the physical Soft Keys count is less than the virtual Soft Keys count, the VT shall provide a means for navigation to allow the operator to choose from any of the Working Sets Soft Keys.

4.5.3.4 Navigation Soft Keys

Navigation Soft Keys is the count of the number of physical Soft Keys that the VT can allocate for the purpose of navigation among the Soft Keys. The number of navigation Soft Keys shall be less than the number of physical Soft Keys. If the VT provides other means of navigation that does not use the physical Soft Keys, this value shall be zero.

4.5.3.5 Navigation among Soft Keys

If the Working Set provides a number of Soft Keys on a Soft Key Mask equal to or less than the number of physical Soft Keys reported by the VT, then all of the Soft Keys on this Soft Key Mask shall be accessible with the physical Soft Keys. The VT shall not provide any navigation means for this Soft Key Mask.

If the Working Set provides more Soft Keys on a Soft Key Mask than the VT has reported in the number of physical Soft Keys, the VT shall provide navigation for that Soft Key Mask. This navigation among the Soft Keys shall be done by paging through the Soft Keys in groups, not by scrolling. Further, a "group" is defined as the "physical Soft Keys" count minus the "navigation Soft Keys" count. The navigation Soft Keys shall always occupy the same physical Soft Key positions on all pages, although the VT designer can choose to disable (but not remove) the navigation keys on certain pages. The last set of virtual Soft

Keys (depending on how many Soft Keys the Working Set provided to the VT) might not completely fill the Soft Key Mask. The remainder of the Soft Key designators shall not be used.

As described in B.5 and illustrated in Figure 4, pointers to the NULL Object ID reserve a Soft Key position. Pointers to NULL Object ID that are at the end of the list of Soft Keys shall not reserve a Soft Key position and shall not be considered for paging or navigation.

EXAMPLE As shown in Figure 4 a), a VT is designed with 6 physical Soft Keys, 64 virtual Soft Keys, and 1 navigation Soft Key. The Working Set provides 18 Soft Keys to the VT, however there are 3 which are Pointers to the NULL Object ID. To support navigating among the Soft Keys the VT designer alters Soft Key 6 into a "next Soft Key group" button. A navigation group is calculated as sets of 5 Soft Keys [Figure 4 a)], starting with the first Soft Key. When the navigation key is pressed, the VT shows the next group of Soft Keys. Another example [Figure 4 b)] shows a similar example with 2 navigation Soft Keys. Another example [Figure 4 c)] shows an arrangement with two columns of keys and two navigation keys. If the VT provides dedicated navigation keys, the number of navigation Soft Keys reported shall be zero [Figure 4 d)].

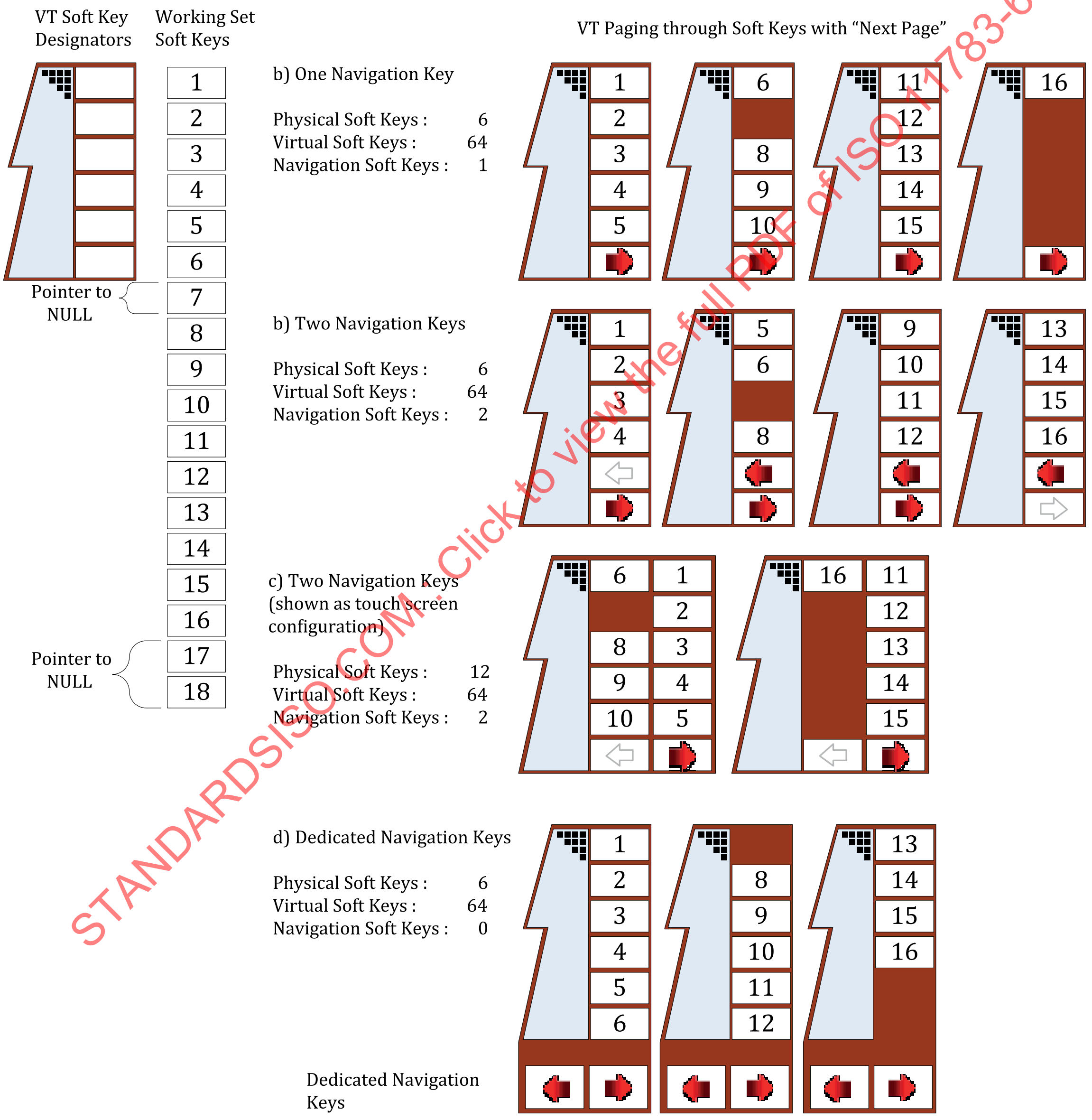


Figure 4 — VT virtual Soft Key paging

4.6 Behaviour

4.6.1 Object pools

4.6.1.1 General

The operator interface definition for a device of one or more implements represented by either a single ECU or a Working Set consists of a set of objects (hereafter referred to as the Working Set's *object pool*).

A summary of the types of objects defining the operator interface definition, the events that can be triggered when interacting with those objects, and a summary of the network commands between a Working Set and the VT are shown in Annex A.

Interaction with the Working Set, caused by the operator interacting with the VT, can be communicated using the messages in Annex H.

During operation, the VT uses other messages not defined within this document, as referred to in Annex I.

These objects are defined in detail in Annex B and Annex J. Each object contains all necessary attributes and child object references for processing the object to completion. The Working Set assigns a unique Object ID to each object in its object pool so that each object is uniquely addressable. Object IDs shall be unique within a single Working Set's object pool but might not be between different Working Sets.

The object pool is transferred to the VT at initialization by using the procedure described in Annex C. The VT is intended to be capable of storing the object pools in a modifiable memory area. VTs can store multiple pools of a Working Set, in non-volatile memory, if they have unique version labels. For example, multiple pools that differ only by language. All objects shall be fully described before they are made active in a mask on the display.

The behaviour of the VT when no object pools are loaded is proprietary.

4.6.1.2 "NULL" Object ID

Object ID FFFF$_{16}$ (65535$_{10}$) is reserved for use as the "NULL" Object ID.

In the context of drawing, and unless otherwise stated, when the "NULL" Object ID is used it means that the VT shall draw nothing for the related object.

4.6.1.3 Processing objects

Objects listed in parent objects can also list child objects, thereby creating a tree hierarchy in the object pool. Objects are always processed in the order listed in the parent object in a "depth-first" manner. In other words, if a reference is made to an object that references other objects, the child references are processed to completion before returning to the parent to continue processing.

VT version 5 and later VTs shall support a minimum hierarchy depth of 30 objects. For VT version 4 and prior VTs, the requirement is unspecified.

The hierarchy depth is computed starting with the following objects; Data Mask object, Alarm Mask object, Window Mask object, Soft Key Mask object, Key Group Objects, and increments by one to reach a child object. For the child objects that have child objects the depth increments again. This process continues to the last child object. For computing the hierarchy depth, the object to which a pointer object references is counted as a child object.

Only contained children are included in the hierarchy count. Referenced objects are not considered in the count (for example a Working Set object referencing a Data Mask object as an attribute).

4.6.2 Working Sets

The Object Pool supplied by a Working Set Master is associated with all members of that Working Set. This allows object information from one CF or all the CFs that make up a Working Set to be collectively presented as a common object pool. One CF shall be designated as the Working Set Master for each Working Set. As coordinator of the communications of a Working Set, the Working Set Master shall secure the use of the VT and provide the object pool definition. It shall also send Working Set messages that provide the NAMEs of the members of said Working Set to the VT. This identifies the members of the Working Set and hence those CFs which can communicate to the VT. Appropriate messages for defining a Working Set are given in ISO 11783-7. If a Working Set Member sends a command or message to the VT that is only allowed from a Working Set Master, the VT shall respond with an Acknowledgement: NACK message (see ISO 11783-3). If a CF is not identified as a member of a Working Set, and where that CF sends a command or message to the VT that is only allowed when from a Working Set, the VT shall respond with an Acknowledgement:NACK message (see ISO 11783-3). Once members of the Working Set have been identified and after the object pool has been loaded into the VT, any member of the Working Set has the ability to provide data for objects and to change attributes in the object pool during run-time. The Working Set Master shall provide the initial object pool definition. Any data input by the operator into input field objects is always transmitted to the Working Set Master.

The VT shall never have Working Set Members and shall not transmit the Working Set Master or Working Set Member messages (see ISO 11783-7).

The handling of VT Response messages defined herein supersedes ISO 11783-1 in reference to responses being directed only to the Working Set Master. See Table 1 for VT Response message behaviour to Working Set messaging.

Table 1 — VT Response message behaviour

Configuration	Working Set Version[a]	VT Version[b]	Behaviour
1	3 and prior	3 and prior	VT response to any command is directed to the WS Master
2	3 and prior	4 and later	VT response to any command is directed to the WS Master
3	4 and later	3 and prior	VT response to any command is directed to the WS Master
4	4 and later	4 and later	VT response to any command is directed to the originator

[a] Working Set Version is reported in the Working Set Maintenance message.

[b] VT Version is reported in the Get Memory response message.

In configurations types 1 through 3, the Working Set Member has the responsibility to monitor all [destination specific] VT to Working Set Master messages in order to pair its commands with responses. The Working Set Master will receive unsolicited responses from the VT (which were originated by its members), and will not be able to pair these with messages the master originated. The Working Set Members also will not be able to pair the messages correctly when originating from another member or from the master.

In configuration type 4, all responses from the VT are directed to the originating nodes. Responses that are communicated via Transport Protocol are now possible (e.g. Get Supported WideChars response). Further, the Working Set Master no longer receives unsolicited response messages. Working Set Members no longer have an obligation to monitor destination specific messages directed to another address.

In order to maintain backward compatibility, Working Sets shall not send higher version messages to a lower version VT (e.g. a version 4 command sent to a version 2 VT). How a lower version VT would respond in such a case should be considered unpredictable. For example some VT designs might respond with an Acknowledgement:NACK message (see ISO11783-3), others might ignore the message. If the connected VT does not support the functional requirements of a Working Set (e.g. VT version 4 features required and the connected VT is version 2, Data Mask size is too small to be functionally useful, etc.), the Working Set shall send a minimally compatible pool to the VT communicating the functional incompatibility to the operator.

Conversely, the VT shall not send higher version messages to a lower version Working Set (e.g. a version 4 event sent to a version 2 Working Set). How a lower version Working Set would respond in such a case should be considered unpredictable. For example some Working Set designs might respond with an Acknowledgement:NACK message, others might ignore the message.

VT version 5 and higher VT's and Working Sets shall support the VT Unsupported VT Function message, and Unsupported VT Function message, respectively. With this message the VT and Working Set respond in a predictable way. VTs and Working Sets, designed for VT version 4 and prior, can implement these messages.

Additional compatibility information is defined in 4.6.24.

4.6.3 Multiple visually similar Working Sets

When more than one visually similar Working Set from the same manufacturer becomes part of a network, these Working Sets should be uniquely identified to correlate each instance with a location. This shall be accomplished using an Instance field of the NAME (e.g. 2 sprayers from the same manufacturer, or 2 or more visually similar Auxiliary Input units).

For consistent system configuration, the Working Sets should be arranged with the lowest to highest instance from left to right followed by front to rear followed by bottom to top.

The manufacturer shall provide the operator a means to establish a working configuration, using one or more of the following methods or via some other means not specified here.

The operator can correctly locate the Working Set based on its instance with

— an indication on the label of the Working Set unit identifying its Instance, and

— by physical location, via a wire in the harness of the Working Set that automatically sets its instance in increasing values left to right followed by front to rear followed by bottom to top.

The operator can set the Instance based on its location with

— an operator accessible “Instance” switch on the Working Set unit,

— an operator accessible “Instance” setting, such as on a Data Mask, and

— use of the commanded name message (see ISO11783-5) with a provided service tool.

4.6.4 Displayed Working Set number

When more than one visually similar Working Set exists on a network, the Working Set shall indicate its working set number on its Working Set object. Additionally, the Working Set should indicate its working set number on visible masks.

The displayed Working Set number shall be defined by the manufacturer. The Working Set number should be related to the Function Instance, the Device Class Instance, and/or the ECU Instance, as defined by the manufacturer (see ISO11783-5). Other factors defined by the manufacturer can also be used to ensure the Working Set is uniquely identifiable. All visually similar equipment from the same manufacturer shall apply the same relationship.

Example 1	working set number = working set Function Instance + 1
Example 2	working set number = working set Device Class Instance + 1
Example 3	working set number = working set ECU Instance + 1

4.6.5 Language, formats and measurement units selection

The VT(s):

- Can query the tractor ECU for the Language command if the VT has no language setting, or if the VT has detected a change in the connected TECU.
- Shall send the standard language, format and measurement units messages defined in ISO 11783-7, hereafter “standard setups”.
- Shall build a superset list of supported languages. The superset list presented to the operator is defined as:
  - the list of all languages communicated to the VT from all Working Set(s), and
  - the list of relevant languages supported by the VT.

  Where the VT supports many more languages than those required by the Working Set(s), it is not required to include the entire list, therefore the languages considered not relevant can be omitted from the superset list presented to the operator.
- Can reduce the list of supported languages when a Working Set disconnects from the VT (see 4.6.9), and previously listed languages are no longer required.
- Shall not store the superset list of supported languages in non-volatile memory.
- The VT shall provide a method for the operator to view the superset list, and to select an item from this list. If the VT itself doesn’t support the language the operator selected it shall provide a proprietary method to switch to an appropriate (or default) language.
- Shall also provide a method for the operator to select formats (Time, Date, etc.) and measurement units. The VT shall report the operator selected language, formats and measurement units at power up and any time there is a change. These messages allow the Working Set to modify its object pool to the operator-selected language (e.g. by updating string fields, selecting units of measure, changing offsets and scales, etc.).
- Shall store the standard setups in non-volatile storage and restore the values during initialization.
- Shall respond to ISO11783-7 “Language Command” requests sent to the global address.
- Shall respond to ISO11783-7 “Language Command” requests directed to this VT.
- Should transmit the Language Command with both the language code (ISO 639-1) and a country code (ISO 3166-1)[14].

The Working Set(s):

- Communicates the initial set of supported languages in the Working Set object.
- Optionally sends the Working Set Special Controls object to supercede the list of languages from the Working Set object which can include both language code and country code.
- Shall configure their standard setups according to the VT to which they are publishing the pool(s). This can cause different standard setups to be published to different VTs. (e.g. auxiliary objects are published to VT with Function instance 0 and the remainder of the pool to other VTs).
- Shall use a proprietary method to select an appropriate (or default) setting if the Working Set does not support the selected language, formats or units.

4.6.6 Initialization

Upon power-up or reinitialization, a specific sequence of events shall occur in order to ensure proper initialization of the VT and Working Sets, as follows.

4.6.6.1 VT initialization

1) The VT shall complete the address claim procedure in accordance with ISO 11783-5 and shall also send an address claim request to the global destination address (255).

2) The VT shall begin transmission of the VT Status message. In the case of a reset or recovery, the VT shall ensure that greater than 3 s have elapsed between this initial VT Status message and the previous VT Status message.

3) If language selection has not been entered by an operator, the VT can attempt to request the default language setting from the tractor ECU, or it can interact with the operator to select a language.

4) The VT shall allow Working Sets to initialize and to load their object pools.

4.6.6.2 Working Set initialization with VT

1) The Working Set, if equipped with Auxiliary Functions, shall clear any assignments in volatile memory.

2) The Working Set Master (and Working Set Members) shall complete the address claim procedure in accordance with ISO 11783-5.

3) The Working Set Master shall wait until the VT begins transmission of the VT Status message.

4) The Working Set Master shall identify itself and its members to the VT using messages given in ISO 11783-7.

   I) The Working Set Master can send these messages for other purposes (e.g. Task controller initialization).

   II) If the Working Set Master has a need to reconfigure the list of Working Set Members after the initialization is complete, the Working Set Master shall send the Working Set Master and Working Set Members messages. The Working Set Master can use this to add or remove members from the set. No Working Set initialization is required.

5) The Working Set Master shall begin the periodic transmission of the Working Set Maintenance message. As identified in Byte 2 and 3 of the Working Set Maintenance message, only the first message can indicate the initiating state.

   NOTE The initiating bit was introduced in Version 3 to provide a clear indication of a Working Set restart.

   If the VT had previously detected a shutdown and was transmitting the Acknowledgement:NACK in response to the Working Set Maintenance message (see 4.6.9), there are two cases where the VT shall stop transmitting the Acknowledgement:NACK:

   I) If the Working Set Maintenance message indicates compliance with VT version 3 and later, and if the initiating bit (Byte 2 Bit 0) is set and the VT received a Working Set Master message since the prior maintenance message, the Acknowledgement:NACK is discontinued.

   II) If the Working Set Maintenance message indicates compliance with VT version 2 and prior, and the VT received a Working Set Master message since the prior maintenance message, the Acknowledgement:NACK is discontinued.

6) The Working Set Master can request the language and format messages from the VT (see ISO 11783-7) if it has not already received this message from the VT and the Working Set has presentation that is language or unit specific.

7) The Working Set Master can query the VT as necessary to determine its capabilities. Based on the VT's responses, the Working Set Master shall adjust its object pool for scaling, available fonts, supported colours, etc.

8) The Working Set Master can query the VT to determine if its object pool already exists in non-volatile memory.

9) Object pool transfer shall commence and be completed. This can be done either by asking for the object pool to be transferred from non-volatile memory (see Annex E) or by using the protocols detailed in Annex C.

10) The Working Set Master can indicate the languages supported by the object pool.

4.6.6.3 Working Set initialization on networks with multiple VTs

A Working Set Master shall have a means to perform a “Move to another VT” function on networks with multiple VTs. This function shall allow movement of the Working Set to each of the available VTs in sequence. For example, this function could be accomplished with a “Next VT” Soft Key or Button in the user interface and/or in combination with the Identify VT message. The function behaves as follows:

1) “Move to another VT” is enabled if the Working Set Master detects more than one VT on the network.

2) When “Move to another VT” is activated, the Working Set Master:

   I) Puts itself in a safe state, or prevents activation of this feature unless it is in a safe state.

   II) Shall send the Delete Object Pool command to the VT and wait for the response.

   III) Shall stop sending the Working Set Maintenance message to the VT.

   IV) Starts the initialization process with another VT on the network.

   V) The Working Set Master shall save the new VT as the preferred VT for a next power cycle. If the preferred VT is not available within a certain time period after startup, the Working Set Master can initialize connection to any other VT on the network. The Working Set can provide a means for the operator to set the maximum wait time period or it can be obtained from the boot time specification in the “Get Hardware response” message of the preferred VT.

4.6.7 System shutdown

4.6.7.1 General

In this context “System Shutdown” is defined as the period of time when the Key Switch state indicates the key is off and yet ECU Power remains on. Actuator Power might or might not remain on concurrent with ECU Power (see ISO 11783-7).

When the Key Switch state indicates the key has been turned off, and while ECU Power has not been terminated, it is expected that systems might transition to a shutdown state that is appropriate for that system. In some devices, this can cause immediate termination of all network communications, where other devices might request power to remain on for a more orderly shutdown. Others still might ignore the Key Switch state and continue normal operation until power is interrupted.

The relevant PGNs defined in ISO 11783-7 for determining the Key Switch state is the Wheel-based speed and distance (PGN 65096) and for requesting power maintenance is the Maintain power (PGN 65095).

The following are recommended practices.

4.6.7.2 VT behaviour

The VT can expect that applications on the network can terminate communications without warning.

A recommended behaviour of the VT is to monitor the Key Switch state and take the following actions as a result of the transition from “Key switch not Off” to “Key switch Off”.

1) The VT should disable unexpected shutdown detection logic to avoid unnecessary notification to the operator as a result of one application shutting down immediately while another maintains the ECU Power beyond the normal 3 s timeout (see 4.6.9).

2) The VT should maintain services while “Key Switch Off” and for a minimum of 2 sec following the last “Maintain ECU Power” request from those ECUs which have object pools in the VT volatile memory.

3) The VT should continue to monitor the Key Switch state and reinitialize if turned from “Key switch Off” to “Key switch not Off”, ensuring that if the VT Status message was discontinued, the standard Initialization process is performed (see 4.6.6).

   NOTE VT version 3 and prior did not specify shutdown behaviour, therefore, these VTs can discontinue all communications with the network, including discontinuing the VT Status message.

4.6.7.3 Working Set behaviour

Working Set behaviour can vary significantly depending on the design of the specific set.

One variation of a Working Set design might not monitor the Key Switch state and might continue as normal until power is lost.

A recommended behaviour of a Working Set is to monitor the Key Switch state and take the following actions as a result of the transition from “Key switch not Off” to “Key switch Off”:

1) The Working Set can send a “Maintain Power” message (see ISO11783-7) to inform the system of the state of the Working Set, and optionally as the means to request power be maintained.

2) The Working Set can monitor the “Maximum time of tractor power” parameter (see Wheel-based speed and distance message in ISO11783-7) and use this information during any power management processes it executes.

3) The Working Set can send a Delete Object Pool command to the VT to eliminate the possibility of an unexpected shutdown indication (see 4.6.9).

4) The Working Set should not consider the lack of the VT Status message or other VT to ECU messages as an unexpected shutdown of the VT, and therefore should not attempt a connection to any other VT that are available.

5) The Working Set should continue to monitor the Key Switch state and reinitialize if turned from “Key switch Off” to “Key switch not Off” (see 4.6.6).

4.6.8 Working Set object and active masks

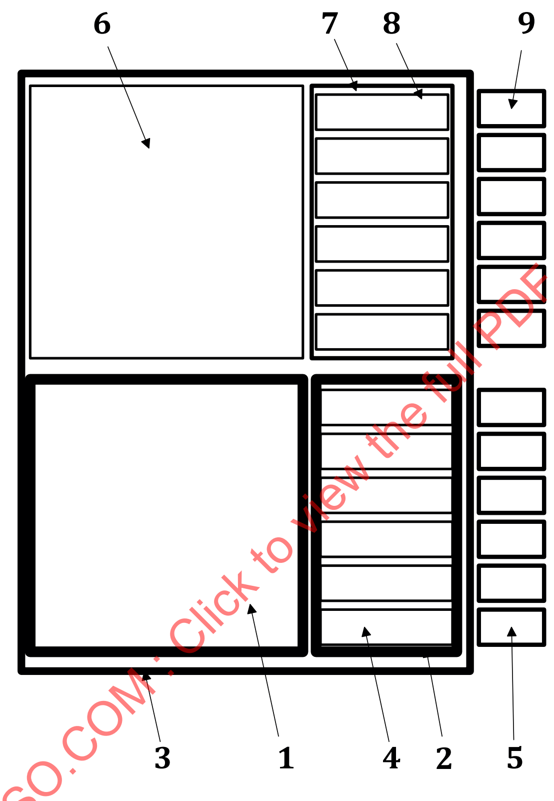
In the initial object pool definition, each Working Set Master shall provide one, and only one, Working Set object in order to define a descriptor, active mask and supported languages for the Working Set. The descriptor can be graphical; text or both but shall fit inside the area defined by the VT for a Soft Key designator. Any object or part of an object located outside of the Working Set descriptor shall be clipped. The descriptor can be used by the VT any time the Working Set needs to be represented to the operator.

EXAMPLE Communication alarms, Auxiliary Control setup.

When a Working Set is “active”, it has exclusive input focus and is displayed on the VT display. When the Working Set is “inactive”, it can also be visible on the VT display but does not have input focus. The VT shall provide some means to allow the operator to select the Working Set that is to be active. Only one Working Set is active at any given time. The Working Set cannot force any of its masks to be visible when the Working Set is not visible, and it cannot force its Working Set to be active when another Working Set is active. Setting the active mask to an Alarm mask shall make the owner Working Set active, unless the active mask of the currently active Working Set is an alarm mask with the same or higher priority (see 4.6.14 Alarm handling).

For VT version 4 and later, a VT can also display one or more Working Sets which are not active in addition to the Active Working set. (See Figure 5, which displays an active and an inactive Working Set simultaneously.) The VT uses the VT On User-Layout Hide/Show message to inform the inactive Working Set to update its Data Mask and or Soft Key Mask when it is visible.

If a Working Set responds with a hidden state for the corresponding Data Mask or Soft Key Mask then the VT knows that the Working Set does not support this feature for this Data Mask or Soft Key Mask. If the Working Set does not support this feature for this Data Mask or Soft Key Mask, the VT shall inform the operator that the displayed information might not be updated. The VT can still display the inactive Working Set, because the inactive Working Set can update its data (see 4.6.10).



Key

1	Data Mask area of active Working Set	6	Data Mask area of inactive Working Set
2	Soft Key Mask area of active Working Set	7	Soft Key Mask area of inactive Working Set
3	physical screen	8	Soft Key Designator of inactive Working Set
4	Soft Key Designator of active Working Set	9	Physical Soft Key of inactive Working Set
5	Physical Soft Key of active Working Set		

Figure 5 — Example VT which displays an active and an inactive Working Set simultaneously

Table 2 — Working Set state changes (VT Supports only Active Mask)

Working Set state change	VT Behaviour
Active to Inactive	1) Hide the Working Set's currently active Data/Alarm Mask and associated Soft Key Mask. 2) Send the VT Status message to the global address (255) to inform Working Sets of the change[a].
Inactive to Active	1) Display the Working Set's currently active Data/Alarm Mask and display the associated Soft Key Mask. 2) Send the VT Status message to the global address (255) to inform Working Sets of the change[1]

[a] When the state of a Working Set changes from inactive to active and this causes the state of another Working Set to go from active to inactive, there shall be only one VT Status message (not two), which shall specify the new active Working Set.

Table 3 — Working Set state changes (VT Supports Multiple Working Sets or Window Masks Visible Simultaneously)

Working Set state change	VT behaviour
Active to Inactive and Visible	1) Remove visual indication that the Working Set is the active Working Set. 2) Send the VT Status message to the global address (255) to inform Working Sets of the change [a]. 3) Send the VT On User-Layout Hide/Show message (state: Shown) to the Working Set.
Inactive and Visible to Active	1) Send the VT On User-Layout Hide/Show message (state: Hidden) to the Working Set (state Hidden is a special case here—refer to H.20). 2) Display the Working Set's currently active Data/Alarm Mask and display the associated Soft Key Mask. 3) Visually indicate to the operator that the Working Set is the active Working Set. 4) Send the VT Status message to the global address (255) to inform Working Sets of the change [a].
Hidden to Active	1) Display the Working Set's currently active Data/Alarm Mask and display the associated Soft Key Mask. 2) Visually indicate to the operator that the Working Set is the active Working Set. 3) Send the VT Status message to the global address (255) to inform Working Sets of the change[a].
Active to Hidden	1) Hide the Working Set's currently active Data/Alarm Mask and associated Soft Key Mask. 2) Send the VT Status message to the global address (255) to inform Working Sets of the change [a].
Inactive and Visible to Hidden	1) Send the VT On User-Layout Hide/Show message (state: Hidden) to the Working Set.
Hidden to Inactive and Visible	1) Send the VT On User-Layout Hide/Show message (state: Shown) to the Working Set.

[a] When the state of a Working Set changes from inactive to active and this causes the state of another Working Set to go from active to inactive, there shall be only one VT Status message (not two), which shall specify the new active Working Set.

The Working Set can select different Data Masks or activate Alarm Masks by changing the active mask attribute of the Working Set object with the Change Active Mask command. The Working Set can change the active mask even if the Working Set is inactive. This allows the appropriate mask to be displayed when the Working Set becomes visible. When a Working Set is inactive, its active mask might not be visible, but still remains as the active mask for that Working Set.

4.6.9 Connection management

The VT transmits the VT Status message once per second. The Working Set uses the message to ensure the VT is present and to determine the current status of the VT. If a Working Set does not receive this message for a period of 3 s it is determined to be a shutdown of the VT. When this happens the Working Set shall enter a safe state. The safe state is defined as the state in which all functions dependant on the VT operator interface are put into a known state that will not put the operator or machine at risk. The Working Set can re-establish connection to the VT. If the Working Set is designed to re-establish connection to the VT it shall do so by restarting the initialization procedure (see 4.6.6).

Each Working Set Master sends the Working Set Maintenance message once per second. The VT uses this message to ensure that each Working Set is still present. If the VT does not receive this message for a period of 3 s or it receives it a second time with the Initiating bit set it is determined to be an unexpected shutdown of the Working Set Master (see Figure 6) and the following rules apply.

The VT shall not alert the operator:

— If the Working Set has commanded the VT to delete the object pool, and the Working Set then stops sending Working Set Maintenance messages. This allows the Working Set to silently remove itself from the VT.

— If the VT can detect the ignition key state and the ignition key is reported as off.

— If there is no pool loaded by the Working Set into the VT volatile memory.

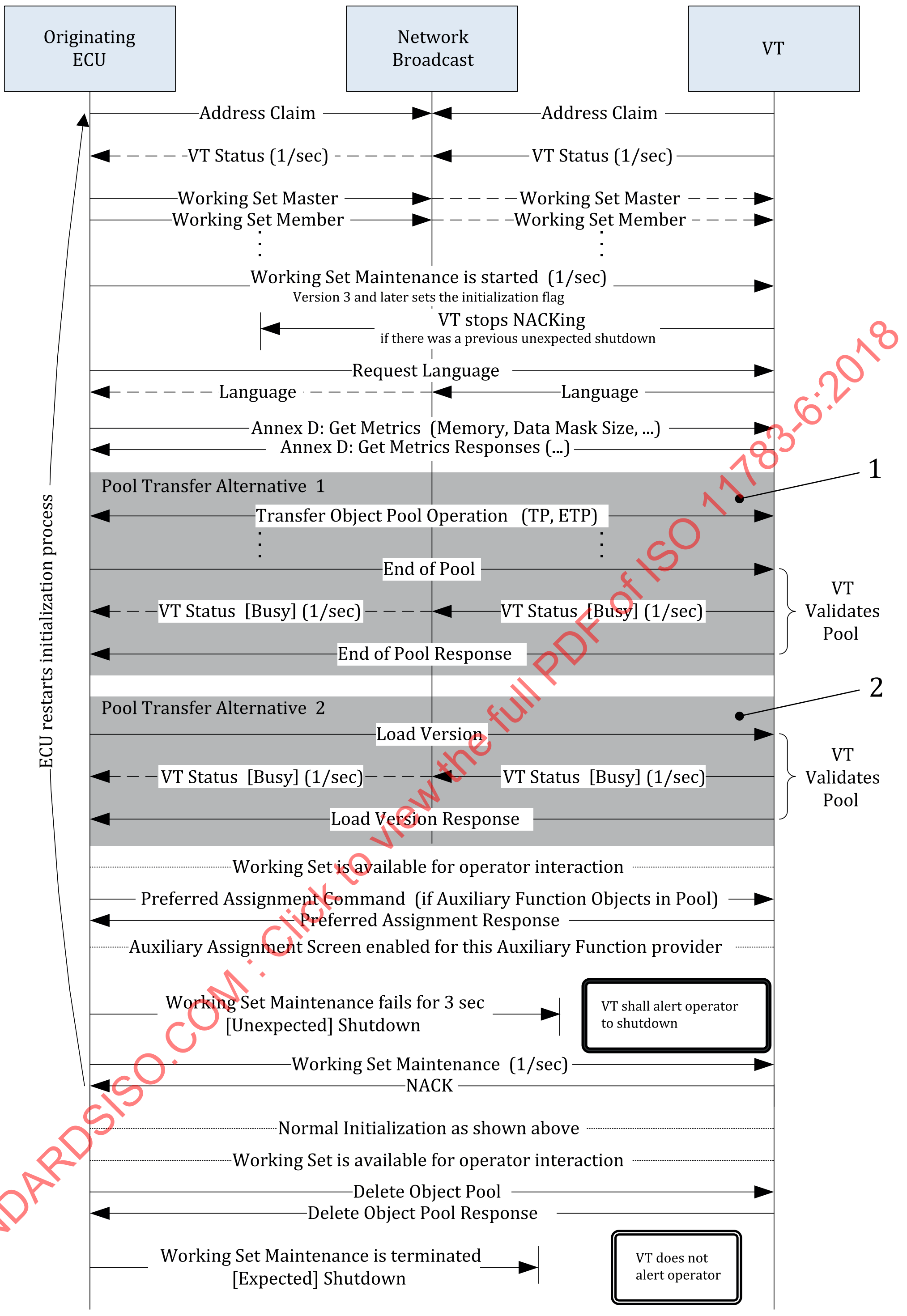
The VT shall alert the operator:

— If the pool has not been commanded to be deleted and the ignition key is not detected as off and the Working Set's object pool is present in the VT. This is detected as an unexpected shutdown of the Working Set and the VT shall alert the operator to this condition after which the VT shall delete the Working Set's object pool from volatile memory to free the memory for other uses. The means to alert the operator is proprietary to the VT.

When a visible Working Set's object pool has been deleted all presentation from that Working Set is removed (Data Masks, Window Masks, etc.). If there was an active Alarm Mask for the removed Working Set the VT deselects the Alarm Mask automatically. The VT can give control to another connected Working Set or to a VT proprietary presentation and it shall update the VT Status message.

When a Working Set's object pool has been deleted and there exists auxiliary assignments mapped to this Working Set, the VT shall remove them.

When the VT receives a Working Set Maintenance message from a Working Set where the initiating bit is not set, and in which the VT had detected an unexpected shutdown, it shall send Acknowledgement: NACK to the message. The Acknowledgement:NACK message is sent to the Working Set Master. The Working Set can re-establish connection to the VT by restarting the initialization procedure (see 4.6.6).


Originating ECU
Network Broadcast
VT
Address Claim
Address Claim
VT Status (1/sec)
VT Status (1/sec)
Working Set Master
Working Set Member
Working Set Master
Working Set Member
Working Set Maintenance is started (1/sec)
Version 3 and later sets the initialization flag
VT stops NACKing
if there was a previous unexpected shutdown
Request Language
Language
Language
Annex D: Get Metrics (Memory, Data Mask Size, ...)
Annex D: Get Metrics Responses (...)
Pool Transfer Alternative 1
Transfer Object Pool Operation (TP, ETP)
End of Pool
VT Status [Busy] (1/sec)
VT Status [Busy] (1/sec)
End of Pool Response
VT Validates Pool
1
Pool Transfer Alternative 2
2
Load Version
VT Status [Busy] (1/sec)
VT Status [Busy] (1/sec)
Load Version Response
VT Validates Pool
Working Set is available for operator interaction
Preferred Assignment Command (if Auxiliary Function Objects in Pool)
Preferred Assignment Response
Auxiliary Assignment Screen enabled for this Auxiliary Function provider
Working Set Maintenance fails for 3 sec
[Unexpected] Shutdown
VT shall alert operator to shutdown
Working Set Maintenance (1/sec)
NACK
Normal Initialization as shown above
Working Set is available for operator interaction
Delete Object Pool
Delete Object Pool Response
Working Set Maintenance is terminated
[Expected] Shutdown
VT does not alert operator
ECU restarts initialization process

Key

1 working Set reboots unexpectedly and within the time-out limit of the Working Set Maintenance message, Object Pool is still in VT volatile memory, Initiating bit is set

2 working Set discontinues communication

NOTE Solid arrows indicate destination specific messaging, dashed arrows indicate global message.

Figure 6 — Initialization, unexpected shutdown, and expected shutdown

4.6.10 Updating the operator interface

4.6.10.1 General

CAN has finite bandwidth available in support of all services (e.g. ISO 11783-3 to ISO 11783-14). Further, the VT has finite bandwidth that is shared by all Working Sets using its services. In order to best manage the system bandwidth, it is recommended that:

— Active Working Sets, or those that are inactive but visible, should issue commands to the VT only when the data has changed in a way which is visible to the operator (e.g. only update objects actively displayed).

— Inactive Working Sets, which have no active Data Mask and Soft Key Mask displayed should reduce the frequency of, or eliminate, updates to the VT.

4.6.10.2 Changing attributes and values

Attributes of objects can be changed during operation by Working Set Masters and Working Set Members using the defined change attribute messages. Certain attributes in each object are assigned an attribute ID. The Change Attribute command allows any attribute with an AID to be changed if not designated as a read-only attribute. In addition, attributes are sometimes grouped together into a single "change" command for efficiency purposes. (e.g. Change Font Attributes command F.28).

Even when the associated Data Mask is not visible, the Working Set can continue to change the attributes (including value) so that when the mask is made active and visible, the necessary output data are current and ready to display.

Objects that are altered by both the Working Set and the operator can be susceptible to a race condition. While the Working Set can know when an object was opened for input (see H.8), processing delays (e.g. inbound and outbound FIFOs) permit the possibility that the Working Set issued a command just as the operator began an interaction. It is the Working Set responsibility to validate commands from the VT.

In VT version 5 and prior the requirements were inconsistent. Some commands provided an error code response "object in use", which is deprecated in VT version 6 and later.

In VT version 6 and later, object attributes can be changed even if the object is "in use". The VT design can apply the change immediately, or the VT design can cache the change until it can be applied. The Working Set can be aware of the current activity (e.g. an object opened for input), and should avoid race conditions that could adversely affect the interaction.

While permissible, changes to specific objects and attributes while "in use" should be avoided:

— Key object; Key code,

— Button object; Key code,

— Input field objects; Value and Variable references,[1], [2]

— Input String object; Input attributes, Extended Input attributes,[1]

— String Variable object; when referenced by an Input String object,

— Input Number object; Minimum, Maximum, Offset, Scale,[1]

1) When a non-atomic means is used to edit Input field objects (e.g. a pop-up dialogue), these attributes shall be transferred to the proprietary edit means as an atomic set, and run-time changes are applied when the edit is completed.

2) The VT Change Numeric Value message has an optional response in VT version 5 and prior. A race condition could occur where the Working Set and VT apply a change, resulting in the VT presentation being out of sync with the Working Set internal value.

— Font Attributes object: font type; when referenced by an Input object,[1)]

— Number Variable object; when referenced by an Input Number object or an Input List object.[2)]

EXAMPLE An Input Number object is being edited with a VT proprietary means such as a pop-up keypad. The Working Set sends a command to change the background colour of the Input Number object being edited. The VT shall accept the command without error, even if the VT design prevents displaying the updated colour while the proprietary editing means is in use. When the editing has completed, and the presentation has been restored where the Input Number object is visible, the applied colour change shall be visible.

4.6.10.3 Changing, adding and deleting objects

Working Sets can replace objects at run-time; however the replaced objects shall be of the same type. New objects can be added by initiating a transport protocol session to send one or more objects to the VT. When the VT receives an object with an existing Object ID, the existing object is replaced (the VT can determine the owner from the source address of the message). Resizing objects is permitted but can cause the VT to run out of memory (see C.2.6).

To delete a single object in the object pool, the entire object pool has to be deleted from the volatile memory in the VT by the Working Set sending a Delete Object Pool command. The object pool has to be uploaded again without the object that should be deleted.

4.6.11 Special objects

4.6.11.1 Container objects

A Container object is a special object used to

— logically group objects in order to identify and reuse the container, or

— hide and show objects.

Figure 7 shows an example of container reuse. Mask 1 and Mask 2 both need the information displayed in Container 1. The Working Set first creates the container, and then inserts the container into Mask 1 and Mask 2 using the Object ID of the container.

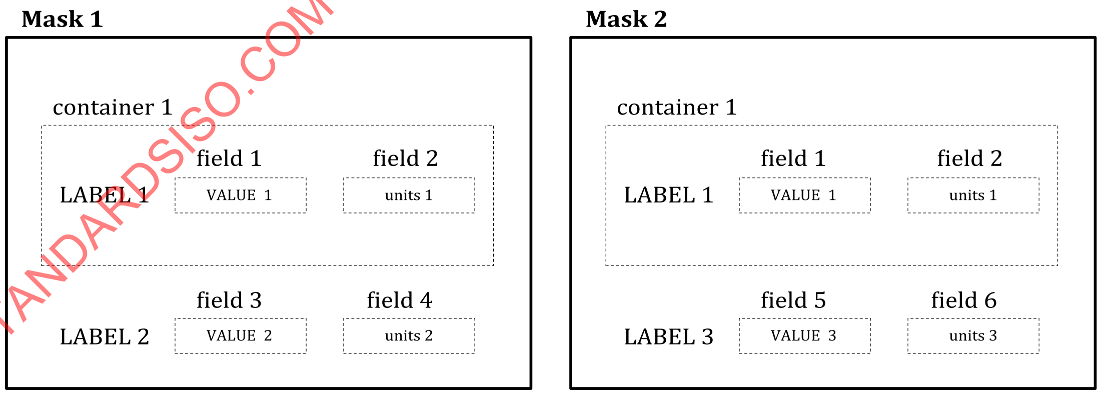
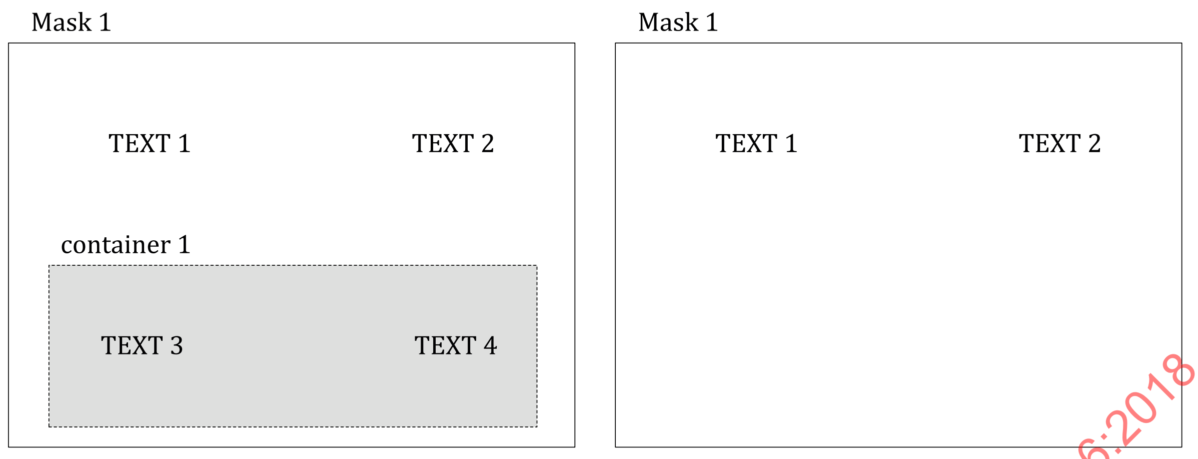


Figure 7 — Container reuse

Figure 8 a) shows an example of using a container to hide a group of objects. In the example, Text 1 and Text 2 should be visible at all times. Text 3 and Text 4 should be visible only if a particular feature of an implement is available. Therefore, the Working Set creates a container containing Text 3 and Text 4 to be inserted in the mask. At run-time, the Working Set determines whether or not the particular feature is available. If not, the Working Set hides the container [see Figure 8 b)].



a) Particular feature available **b) Feature not available: container hidden**

Figure 8 — Container used to hide objects — Example

4.6.11.2 Attribute objects

There are five types of attribute objects: font, line, fill, input, and extended input. Attribute objects are referenced by other objects. This allows one set of attributes to be shared with many objects to create and maintain a common look across those objects. All objects using a given attribute object are updated when the attribute object is changed.

4.6.11.3 Variable objects

Variable objects can be used to share data between two or more other objects. Changing the value in only one object can reduce bus traffic. For example, it could be desirable to draw a Meter object and also to show its current value as a numeric under the meter. In this case, a Number Variable object could be referenced by both the Meter object and an Output Number object. By doing this, a change in the value of the variable will be reflected in both the meter and output field at the same time and the change will require only a single Change Numeric Value command.

4.6.11.4 Macros

Macro objects are used to improve the performance of the operator interface. Macro objects have the following properties:

a) Macros can only contain the commands listed in Annex F.

b) If a Macro triggers an event that causes another Macro, the current Macro is completed first before another is started.

c) Macros are executed in the order they were triggered.

d) If a Macro executes an Execute Macro command or an Execute Extended Macro command, the macro in the Execute command is executed, before the remaining commands in the current Macro.

e) Macros triggered by the execution of a command shall be completed before the next bus command is started.

f) Macro Object IDs shall be in the range 0 to 255 for VT version 4 and prior. Macro Object IDs can be in the range of 0 to 65534 for VT version 5 and later.

g) Macros do not trigger response messages. When executing the command messages in a macro, the VT shall not send response messages on the CAN bus for the messages contained in the macro.

Therefore macros reduce CAN traffic. For example, a Macro that executes a Change Active Mask command will trigger a VT Status message, but will not trigger a Change Active Mask response.

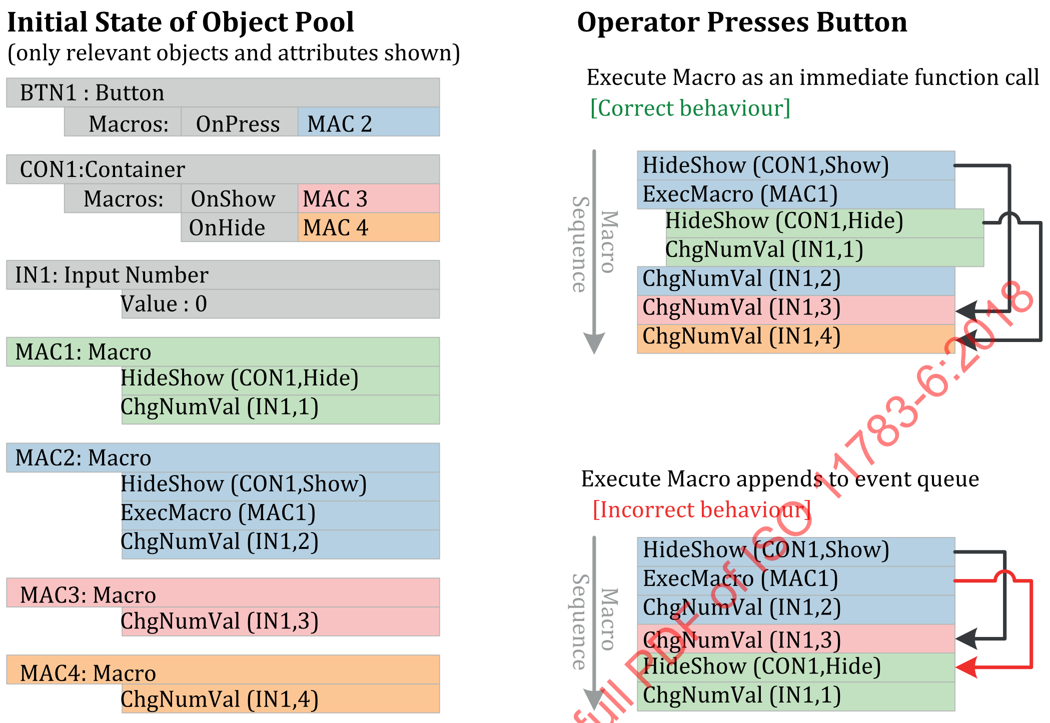


Figure 9 — Macros; The effect of Execute Macro in a Macro — Example

As can be seen in the example of Figure 9, the correct execution of this sequence results in an Input Number containing the value 4.

CAUTION — Objects that are altered by both a Working Set and a Macro can be susceptible to a race condition and should be evaluated for predictable behaviour.

EXAMPLE 1 An Input String object has a Macro to clear the Input String object—On Input Field Deselection. The "Enter" means on an Input String object of length greater than 3 characters causes the VT to send the data using Transport Protocol (TP) to the Working Set (WS). The operator quickly navigates away from the Input String object. "On Input Field Deselection" causes the VT to notify the WS with a VT Select Input Object message (Deselect). An On Input Field Deselection Macro alters the Input String object value. The WS receives the TP data asynchronous to the VT Select Input Object message (Deselect). The outcome is based on the asynchronous processes in the VT, the message processing methods in the VT, including TP processing and potential latencies, and the WS implementation.

EXAMPLE 2 An On Key Press Macro associated with a Button object changes the active Data Mask. An On Show Macro attached to the new Data Mask changes the numeric value in an Input List object on this new Data Mask to initialize it to a known setting. The operator presses the Button, which causes a Button Activation message to be sent to the WS. The WS sends a Change Numeric Value command to the Input List object. The Input List object can end up with either value, due to the asynchronous processes.

Circular references are not permitted since they create infinite Macro loops inside the VT and could render the VT inoperable for all Working Sets.

EXAMPLE 3 When a Macro triggers an event that references the same Macro, a circular reference is created.

4.6.11.5 Object pointer

The Object Pointer object allows run-time modification of included objects. By changing the value of the Object Pointer object, a different object can be drawn at the same location. The type of object that the Object Pointer is allowed to reference is limited and depends on the parent object. Refer to valid parent objects of the Object Pointer for a list of objects that can be referenced. An Object Pointer can always point to another Object Pointer. An Object Pointer can point to the NULL Object ID and in this case nothing shall be drawn.

When an Object Pointer is modified at run-time, resulting in an invalid object reference, the VT is not required to detect this object pool error immediately but can delay error detection until the Data Mask object or Alarm Mask object which contains the Object Pointer is activated. At activation of a data or Alarm Mask containing the invalid object reference the VT sends the F.35 "Change Active Mask response" or H.14 "VT Change Active Mask message" to inform the Working Set and can delete the object pool.

The VT can also detect the error immediately upon Object Pointer value modification, and in this case shall send an F.23 Change Numeric Value response with "invalid value" indicated in the error codes.

4.6.11.6 External Object Pointer

The External Object Pointer object allows a WS to display objects from an object pool of another WS.

To ensure that the external references are valid even after software updates of the participating WS, the referenced WS and the referencing WS shall exchange information about the referenced objects before enabling the external references. The information exchange shall be repeated every time either the referencing WS or the referenced WS is restarted. As a minimum the object identifiers shall be transferred from the referenced WS to the referencing WS.

NOTE Some devices (e.g. the SCC/SCM — ISO 11783-14) have a standardized method for exchanging object information. If a standardized method does not exist, then a proprietary method shall be agreed between the manufacturers of the referencing and the referenced WS.

To protect against unsolicited references the referenced WS shall use the External Object Definition object to list the objects which are allowed to be referenced by another WS. The External Object Definition object is assigned to one and only one referencing WS. If a referenced WS will allow multiple WS to reference objects it shall have multiple External Object Definition objects.

The referencing WS shall use the External Reference NAME object to identify the WS it plans to reference.

The attribute values of the External Object Definition object, the External Object Pointer object and the External Reference NAME object shall all be valid before the external object can be displayed. Some of the information in the objects depends on the exchange of object information and therefore the objects shall be in the reset state until the object information is complete.

Reset state is defined by:

— External Object Pointer object	: External Object Id attribute is the NULL object id
— External Object Definition object	: Enable bit in the Options attribute is cleared
— External Reference NAME object	: Enable bit in the Options attribute is cleared

When the object pool is loaded from non-volatile storage in the VT there is no guarantee that the attribute values are valid, and therefore the VT shall set all occurrences of the External Object Pointer, the External Object Definition and the External Reference NAME to the reset state.

When the object pool is uploaded from the WS master, the VT shall not set the objects to the reset state. If the information exchange between the referenced and the referencing WS is not completed before the object pool is uploaded, the WS master shall set the objects to the reset state before uploading the pool.

Before displaying an external object the VT shall check the validity of the external reference. The external reference is considered to be valid when all of the following are fulfilled:

— The External Reference NAME ID attribute of the External Object Pointer object identifies an enabled External Reference NAME object.

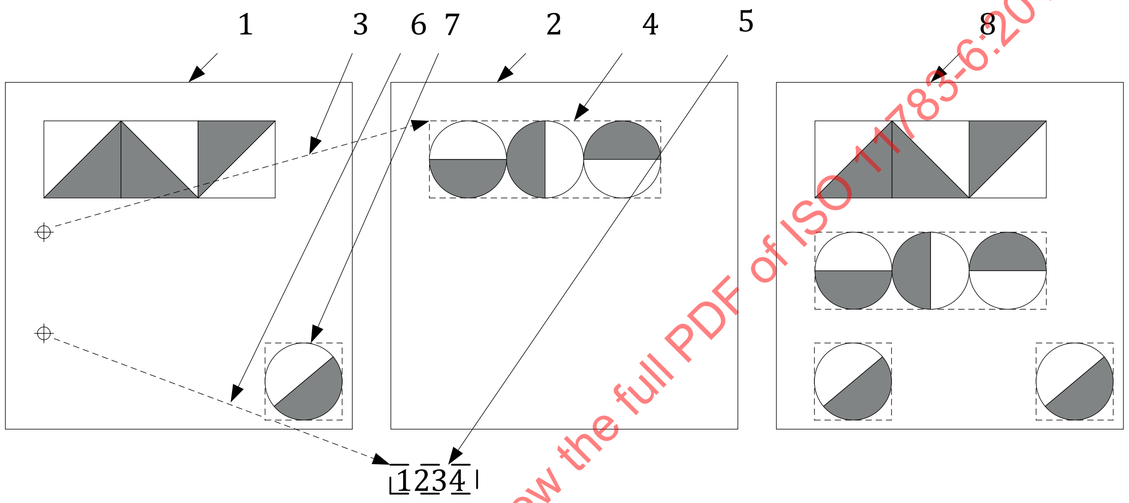
— The referenced WS has an object pool in volatile memory on the VT (referenced object pool).

— The referenced object pool contains an enabled External Object Definition object where the NAME attributes identifies the referencing WS.

— External Object Id is listed in the object list in the above mentioned External Object Definition object.

If the referenced object is NULL or not valid the VT shall draw the object identified in the Default Object ID attribute of the External Object Pointer object.

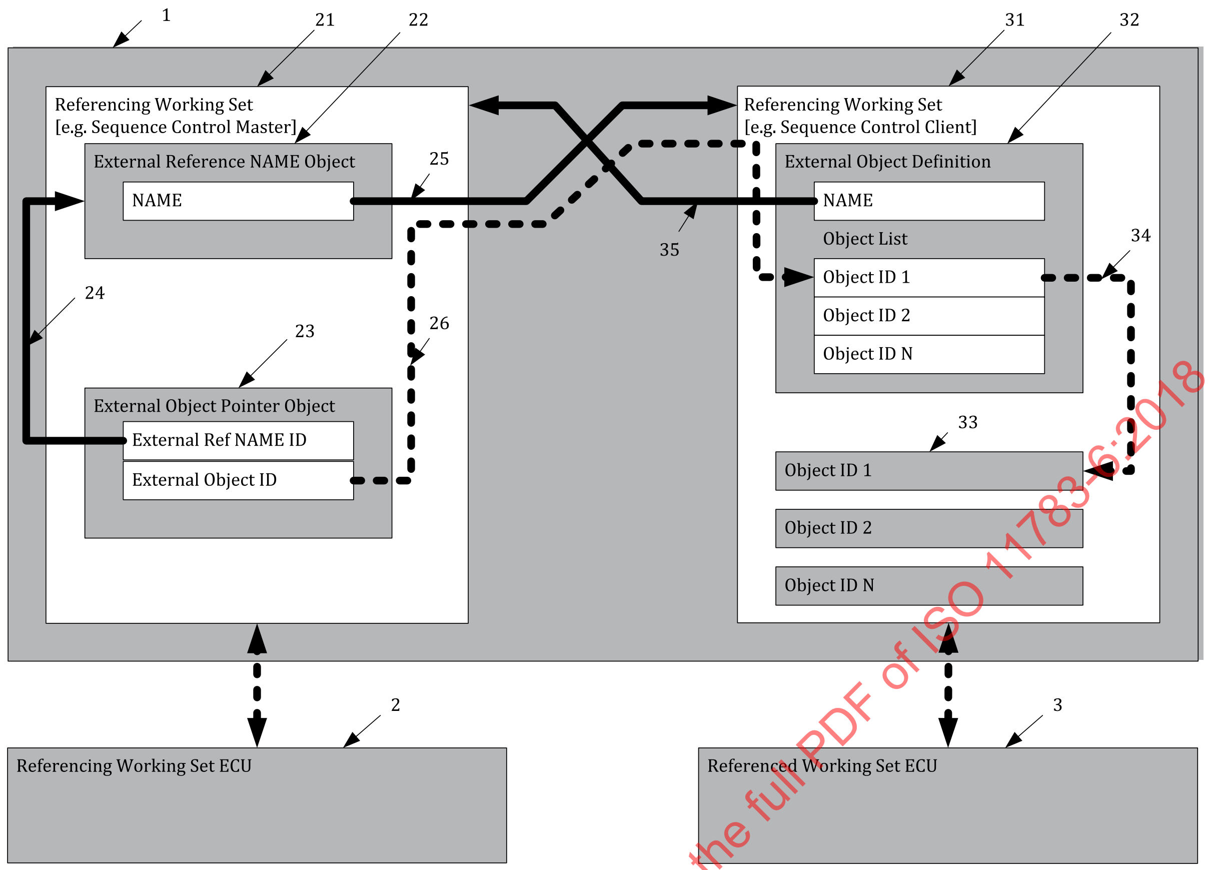
There can be multiple instances of the External Object Definition object in an object pool. If multiple External Object Definition objects are assigned to the same NAME, then an external reference shall be considered valid if made valid by one or more External Object Definition objects.



Key

1 VT presentation from Working Set 1
2 VT presentation from Working Set 2
3 External Object Pointer with reference to object in Working Set 2 Object Pool
4 Referenced object in Working Set 2 (container that contains several objects), all of which are valid
5 Number Variable on Working Set 2 object pool
6 External Object Pointer with reference to object in Working Set 2 Object Pool
7 Default Object to be shown when External Objects are invalid
8 VT presentation from Working Set 1 after references are established, enabled and resolved

Figure 10 — External Object References — VT Example


1
21
22
31
32
Referencing Working Set
[e.g. Sequence Control Master]
External Reference NAME Object
NAME
25
24
23
26
External Object Pointer Object
External Ref NAME ID
External Object ID
Referencing Working Set
[e.g. Sequence Control Client]
External Object Definition
NAME
Object List
Object ID 1
Object ID 2
Object ID N
35
34
33
Object ID 1
Object ID 2
Object ID N
2
3
Referencing Working Set ECU
Referenced Working Set ECU

Key

1	VT Object Pool volatile memory
2	Referencing Working Set ECU (ECU 2)
3	Referenced Working Set ECU (ECU 3)
21	Referencing Working Set in VT memory
22	External Reference NAME Object, including NAME attribute that references ECU 2s
23	External Object Pointer Object
24	Reference to the External Reference NAME object
25	Virtual reference established by External Object Pointer object that informs VT to the Working Set containing the referenced objects
26	Virtual reference established to the External Object of interest
31	Referenced Working Set in VT memory
32	External Object Definition object
33	Object Pool Object that can be referenced from an external Working Set
34	Reference to the Object Pool object that can be referenced
35	Virtual reference established that allows ECU 2 to reference ECU 3 objects

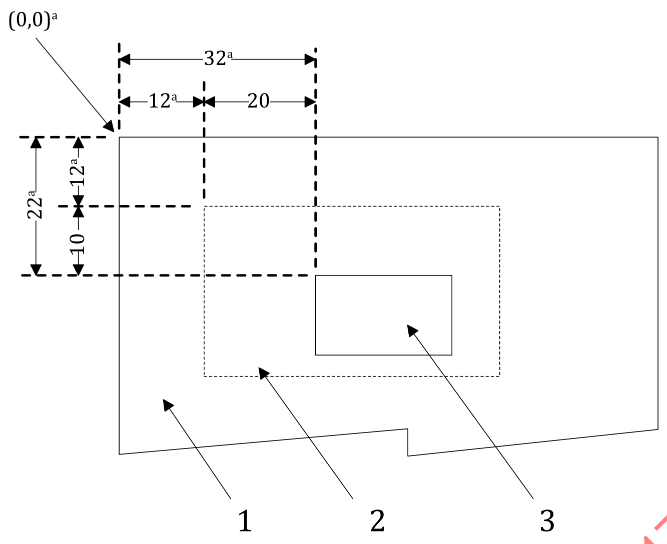
Figure 11 — External Object References — Relationship Example

Example 1 Two External Object Definition objects have the same value in the NAME attributes. Both objects are enabled, only one of the objects includes 1234 in the Object ID list. Object Id 1234 can be referenced.

Example 2 Two External Object Definition objects have the same value in the NAME attributes. Both objects include 1234 in the Object ID list, only one of the objects is enabled. Object Id 1234 can be referenced.

4.6.12 Relative X/Y positions

The X, Y position attribute determines where an object is drawn on the display. This position is always relative to the upper left corner of the parent object. The X,Y position is always found in the parent object. Figure 12 shows an example Data Mask with the relative locations of several objects.



Key

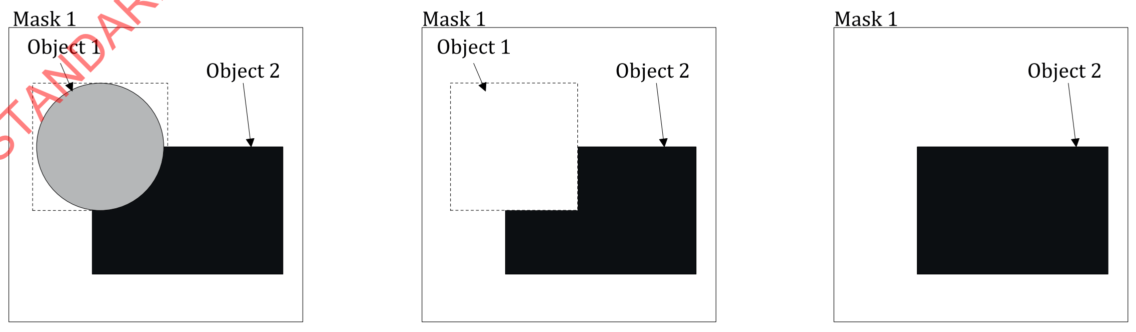
1	data mask
2	container
3	input field
a	Absolute.

Figure 12 — Relative and absolute location of objects

4.6.13 Overlaid objects

A mask can be built such that two objects will occupy the same or overlapping space on the display. This can require extra processing in the VT, but could be necessary in some cases and shall be supported by the VT. For this reason, the hierarchy or layering of objects shall be understood. Some objects have a list of contained objects. Objects listed first are considered to be lower in the hierarchy than objects listed later. When an object is changed, all objects that overlay it and which have been corrupted shall be redrawn (this is called a refresh event). All objects defined by this part of ISO 11783 have a rectangular size either defined or implied to simplify the VT's task of finding overlaid objects.

Objects shall be drawn such that objects listed later appear to overlay objects listed earlier, so that the entire image of the object listed last is visible, while only those areas of the object listed earlier that do not coincide with areas of the object listed last will be visible.

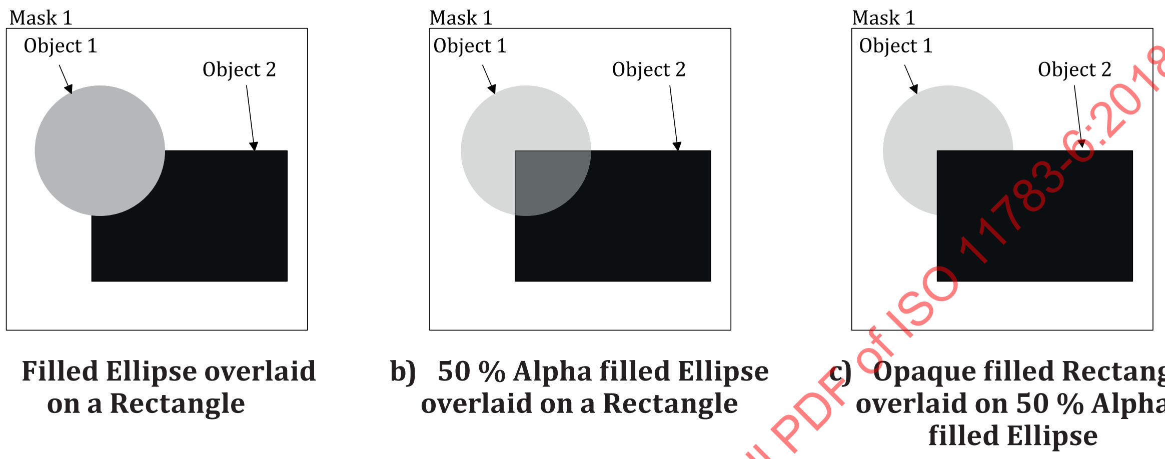


a) Initial presentation **b) Intermediate operation** **c) Final presentation**

Figure 13 — Object changed or hidden — Display update

EXAMPLE Referring to Figure 13, where (a) shows the initial presentation before any command is received. The Working Set then commands object 1 to be hidden. As an intermediate step shown in Figure 13 b), the VT deletes object 1 and all child objects by filling the object's area with the background colour of the parent mask. The VT then redraws the object along with all child objects. The VT then refreshes any other object (e.g. object 2) that could have been visually altered in the deletion process as shown in Figure 13 c). The VT can implement this refresh in a manner where the intermediate display is never seen by the operator.

Additionally, VT version 6 and later supports redefinition of the VT standard colour palette with one that includes alpha channel (transparency) values. This causes alpha blending in the area of the overlaid objects when the top object is defined with a non-opaque alpha channel.



a) **Filled Ellipse overlaid on a Rectangle**

b) **50 % Alpha filled Ellipse overlaid on a Rectangle**

c) **Opaque filled Rectangle overlaid on 50 % Alpha filled Ellipse**

Figure 14 — Overlaid Objects including Alpha channel effects

EXAMPLE Referring to Figure 14, where a) shows the presentation when an Ellipse is filled with an opaque colour, Figure 14 b) shows the presentation when the fill colour has been redefined with a 50 % alpha value, and Figure 14 c) shows that an opaque object on top of an object with a 50 % alpha value does not cause blending.

Alpha channel values range from fully transparent (0) to fully opaque (255). Where the alpha channel information affects the display, the blended value is defined as follows:

— Source colour is the colour of the top object (the object listed later).

— Destination colour is the rendered colour of the underlying (bottom) object (the object listed earlier), which can have been blended with other underlying objects.

— Blended colour = (source colour × alpha/255) + [destination colour × (255 − alpha)/255][3].

Two or more objects can be positioned in an overlay. Starting from the bottom-most object each blended colour layer shall be computed individually, thus defining the destination colour for the next object in turn.

Referring to Figure 14, the background is opaque white RGB(255,255,255), the rectangle is black RGB(0,0,0), and the ellipse is grey RGB(128,128,128). In drawing b, the ellipse is 50 % Alpha A(128). Where the ellipse overlays the white background, the blended colour is RGB(191,191,191) and where the ellipse overlays the black rectangle, the blended colour is RGB(64,64,64).

4.6.14 Alarm handling

Alarms allow a Working Set to display alarm information at any time. If several Working Sets have an Alarm Mask activated, the VT shall display the masks in order of priority. Priority is determined, first, by the priority attribute defined in the Alarm Mask object and, second, by chronological order of activation. The highest priority alarm is always displayed until the owner Working Set changes the active mask. When more than one Working Set has activated an Alarm Mask with the same priority attribute, the first of these, as processed by the VT, shall become the active mask. When the active

3) The algorithm employed can round or truncate in the production of the Blended colour.

Working Set changes to a lower priority alarm or Data Mask, the next in turn highest priority Alarm Mask is processed.

When the Working Set and the VT are compatible with VT version 6 and later (see D.3, G.3), A Working Set that is not selectable can activate an Alarm Mask (using the Change Active Mask command). To deactivate the Alarm Mask, the Working Set would activate a Data Mask. The VT shall apply normal Alarm Handling rules.

Operator input disturbed by the activation of an Alarm Mask can be left intact for resumption once all Alarm Masks have been acknowledged. A Soft Key Mask is associated with the Alarm Mask via an attribute in the Alarm Mask object. Whenever the alarm is displayed, the associated Soft Key Mask is also displayed. The following describes the protocol requirements between the VT and the Working Set raising the alarm.

a) The Working Set Master activates an Alarm Mask by using the Change Active Mask command on the Working Set object. Only one Alarm Mask can be active per Working Set.

b) The VT responds with a Change Active Mask response.

c) Based on priorities, at some point, the VT displays the Alarm Mask and the Soft Key Mask associated with the Alarm Mask. When a mask change occurs that causes an Alarm Mask to appear or reappear, the acoustic signal associated with the Alarm Mask shall be activated. The VT shall terminate any acoustic signal from a lower priority Alarm Mask that can be in process, even if the higher priority Alarm Mask is set for silent. The VT shall terminate any acoustic signal from a Control Audio Signal command that can be in process if the VT does not support multisound (see F.10) and if the acoustic signal is not set for silent (version 4 and later VTs shall send the VT Control Audio Signal Termination message to indicate the termination). If the Alarm Mask acoustic signal completes, or if it is set to none for silent signal, then control audio commands from ECUs shall be accepted as defined in the Control Audio Signal command (see Table 2 and Table 3).

d) The VT notifies the Working Set with a VT Status message sent to the global address.

e) At some point the operator can acknowledge the alarm with the proprietary ACK means on the VT. The VT sends a Soft Key Activation message to the Working Set with key code set to zero (0) based on the operator action (see 4.6.18). Alternately, the Working Set can deactivate the Alarm Mask using the Change Active Mask command to activate a Data Mask; based on its internal logic.

f) The Working Set can choose to ignore the ACK means if the ACK is not allowed or the Working Set can change the active mask to either an Alarm Mask or a Data Mask by using the Change Active Mask command on the Working Set.

g) The VT responds with the Change Active Mask response.

h) The VT displays a new mask (see Table 4).

Table 4 — VT behaviour on mask transition

Working Set's active mask attribute From/To	Is requester the Current active Working Set?	VT behaviour
Data to data	Yes	Hide current Data Mask, show new Data Mask.
Data to data	No	If the VT design supports only one visible Data Mask, then no visual change. If the VT design supports multiple visible Data Masks and this Working Set is currently visible, then hide the current Data Mask, show new Data Mask.
Data to alarm	Yes	Hide Data Mask, show Alarm Mask.
Data to alarm	No	If this is the highest priority alarm, deactivate the current Working Set and activate this Working Set.

Table 4 *(continued)*

Working Set's active mask attribute From/To	Is requester the Current active Working Set?	VT behaviour
Alarm to alarm	Yes	If this is the highest priority alarm, hide current Alarm Mask, show new Alarm Mask. Otherwise, deactivate this Working Set and activate the Working Set of the highest priority alarm.
Alarm to alarm	No	If this is the highest priority alarm, deactivate the current Working Set and activate this Working Set.
Alarm to data	Yes	If an alarm exists in another Working Set, deactivate this Working Set and activate the Working Set with the highest priority alarm. Otherwise, if this Working Set had the last visible Data Mask, hide the Alarm Mask and show the Data Mask. Otherwise, deactivate this Working Set and activate the Working Set which had the last visible Data Mask. If there is no Working Set which had the last visible Data Mask, the VT presentation shall be the same as when there is no Working Set mask to display.
Alarm to data	No	No visual change.

4.6.15 Clipping

Most objects defined in this part of ISO 11783 have a given or implied size. The VT shall clip anything drawn outside the defined size of the object. Clipping is always done on a graphical (i.e. pixel) basis.

These clipping rules also apply to text and numeric objects. When the text does not completely fit inside the defined object area, in both wrapping and non-wrapping cases, the graphical clipping rules apply and the presentation is clipped on a graphical (i.e. pixel) basis (see Figure 15).

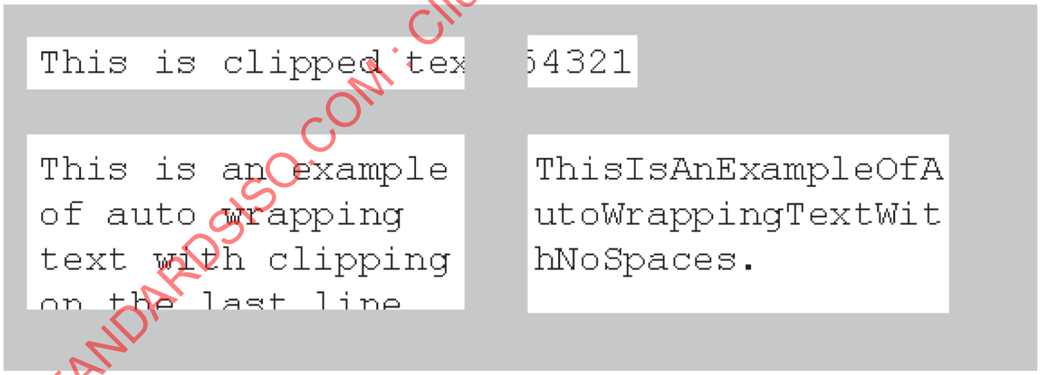


Figure 15 — Clipping examples

4.6.16 Scaling

4.6.16.1 General

The Working Set shall determine the size of the VT's Data Mask area and Soft Key designator and make appropriate adjustments to its object definitions. These adjustments can be applied either before or after transmission of its object pool to the VT, as long as the transmitted pool is not invalid for the VT (e.g. cannot transmit colour objects to a black and white VT and then change object to black and white). This gives complete control of the appearance of the masks to the Working Set.

4.6.16.2 Positions and sizes

The Working Set shall scale positions and object sizes to adapt to the VT's Data Mask area and Soft Key designator(s).

4.6.16.3 Fonts

The Working Set shall apply a best-fit algorithm to determine and select the best font for the defined area. The Working Set shall also ensure that the VT supports the selected fonts and font styles. The smallest font size is 6 × 8. If the scaled height and width of the original font is less than 6 × 8, the 6 × 8 font shall be used. This could result in clipping if the text is near the edge of the field object, or in text overlap if two occurrences of text are too close together after scaling the object size independently of the font size. Working Set designers shall be aware of this limitation and shall take the necessary steps. (See 4.6.19.3 and 4.6.19.4.)

4.6.16.4 Picture graphic objects

Picture Graphic objects are automatically scaled by the VT according to the width attribute in the Picture Graphic object.

4.6.17 Operator input

Whenever the VT displays a Data Mask that contains one or more enabled visible input objects or Buttons, it shall be in one of the following states:

a) Navigating;

b) Data input[4].

When determining the visibility of an input object, the object shall be considered visible even under the following conditions:

— the object's width or height equals zero;

— the object is covered in its entirety by another object;

— the object is wholly outside the clipping limits of the parent object hierarchy as defined by the width and height attributes of all the parents in the hierarchy.

Objects with width or height of zero, and objects that are wholly outside the clipping limits of the parent object hierarchy are discouraged as activation might not be possible (e.g. touch screen). The Working Set designer can consider enabling and disabling these objects as needed so the Working Set navigation is more predictable in operation.

When a new active data-mask is selected the state is reset to "Navigating". If an Alarm Mask is selected then the VT might remember the state and if the Working Set returns to the same data-mask after showing the alarms then the state can be restored. If this approach is implemented and the Working Set returns to a different Data Mask after showing alarms, the VT shall consider it as a normal Data Mask change and send the appropriate messages (see Table 5).

The initial focus point, if any, and the tab order for navigating to the various input fields, buttons or soft keys is VT proprietary, but the Working Set shall be aware that the tab order can be defined by the definition order of the input objects in the parent object.

NOTE VT version 3 and prior do not support selection of a Button object or a Key object using the VT Select Input Object message.

The VT can choose not to send the VT Select Input Object message for every object that the operator passes through while navigating. E.g. if the navigation means is a rotary control, the focus will change

4) The nature of buttons means that the VT cannot be in the Data input state when a button has focus.

rapidly while the operator is spinning the control. In such a case the VT can choose only to send the VT Select Input Object message to the input object which loses focus and the one which eventually gets focus.

In the “Navigating” state the VT shall indicate to the operator which (if any) input, Button, or Key object is selected (has focus). The method of indication is proprietary to the VT. The VT can open an input object for data input as soon as the object gets focus, and it can remove focus when input is done. This is common, but not required, behaviour for touch screens.

Examples of focus-indicators include (but is not limited to) adding a frame around the input field, changing the background colour of the input field, or momentarily highlighting the input field on a touch screen VT.

The VT designer should be aware that the use of the Select Input Object command by the Working Set can be a means by which the Working Set designer intends to focus the operator attention to an input field (e.g. a setup wizard where the recommended action is highlighted).

The VT shall indicate the disabled input objects. The means to represent disabled input objects is VT proprietary. Visual changes to disabled objects to indicate the disabled state shall not extend beyond the width/height of the object and the object shall remain legible.

Working Sets can apply a frame around, or distinct background colour to, input objects as an aid to the operator in identifying input fields.

In the “data input” state the VT behaviour is proprietary, and the VT can cover part or all of the Data Mask while the object is open for data input. Changes to the attributes of an input object during the data input process shall not affect the value currently being input (e.g. changes to an Input Number Scale shall not alter the apparent value being input).

The VT reacts to navigation related events (see Table 5). The VT shall respond with all messages indicated in Table 5 for each specific scenario. The sequence order of the response messages in Table 5 is not a requirement and Working Set Masters shall be designed to handle the responses in any order.

Table 5 — VT Reaction to navigation and data input events

Current state	Command/Event	New state	Response frames
Navigating	Select Input Object command (byte 4 = FF_{16})	Navigating	Select Input Object response
Navigating	Enable/Disable Object command (to disable the object which has focus)	Navigating	(The object becomes disabled and loses focus—VT can move focus to the next input object) Enable/Disable Object response VT Select Input Object message (on object which loses focus) VT Select Input Object message (on object which gets focus — if any)
Navigating	Change Active Mask command (only if new mask is a new Data Mask, or if the new mask is an Alarm Mask and the VT does not store the state of the input object)	Navigating	VT Select Input Object message (on object which loses focus — if any) Change Active Mask response VT Status message (with new Data Mask — if received from the active working set) VT Select Input Object message (on object which gets focus — if any)

Table 5 *(continued)*

Current state	Command/Event	New state	Response frames
Navigating	Select Active Working Set command	Navigating	VT Select Input Object message (on object which loses focus — if any) Select Active Working Set response VT Status message (with new WS and Data Mask) VT Select Input Object message (on object which gets focus — if any)
Navigating	ESC command	Navigating	ESC response (with error code indicating that no input is open for input)
Navigating	Operator activates a selected Button object	Navigating	Button Activation message
Navigating	Operator navigates to a new object	Navigating	VT Select Input Object message (on object which loses focus — if any) VT Select Input Object message (on object which gets focus — if any)
Navigating	Operator opens the object for data input	Data input	VT Select Input Object message (on object which has/gets focus)
Navigating	Select Input Object command (byte 4 = 00_{16})	Data input	Select Input Object response
Data input	Select Input Object command (Selecting an object that does not currently have focus)	Data input	Select Input Object response (with error code indicating that another input field is currently being entered)
Data input	Enable/Disable Object command (to disable the object which has focus)	Data input	(the input object stays enabled and maintains focus) Enable/Disable Object response (with error code indicating that operator input is active)
Data input	Change Numeric Value command (on the object that has focus)	Data input	Change Numeric Value response
Data input	Change String Value command (on the object that has focus or the Input Attribute that is referenced by the object that has focus)	Data input	Change String Value response
Data input	Change Active Mask command (only if new mask is a new Data Mask, or if the new mask is an Alarm Mask and the VT does not store the state of the input object)	Navigating	VT ESC message VT Select Input Object message(on object which loses focus) Change Active Mask response VT Status message (with new Data Mask) VT Select Input Object message (on object which gets focus — if any)

Table 5 *(continued)*

Current state	Command/Event	New state	Response frames
Data input	Select Active Working Set command	Navigating	VT ESC message VT Select Input Object message (on object which loses focus) Select Active Working Set response VT Status message (with new WS and Data Mask) VT Select Input Object message (on object which gets focus — if any)
Data input	Change Attribute command (of the object that has focus)	Data input	Change Attribute response
Data input	Change List Item command (on the object that has focus)	Data input	Change List Item response
Data input	Pool Update alters the object which has focus	Navigating	VT ESC message VT Select Input Object message (on object which loses focus — if any) VT Select Input Object message (on object which gets focus — if any)
Data input	Parent Container is hidden	Navigating	VT ESC message VT Select Input Object message (on object which loses focus — if any) VT Select Input Object message (on object which gets focus — if any)
Data input	Pointer to this object is changed to not reference this object	Navigating	VT ESC message VT Select Input Object message (on object which loses focus — if any) VT Select Input Object message (on object which gets focus — if any)
Data input	ESC command	Navigating	ESC response VT Select Input Object message (on object which has/loses focus)
Data input	Operator activates the ESC means	Navigating	VT ESC message VT Select Input Object message (on object which has/loses focus)
Data input	Operator activates the ENTER means	Navigating	VT Change Numeric Value message or VT Change String Value message (even if the new value is the same as the old) VT Select Input Object message (on object which has/loses focus)

4.6.18 Soft Key and Button activation

Whenever a Key object or Button object or ACK key is pressed, released, or latched, the VT sends a Soft Key Activation message or Button Activation message to the working Set Master. If a Macro is associated with the key press, the VT executes it. Performance of the operator interface can be improved by associating a Macro with the key press event to cause another event, such as activating a different mask. See Table B.11 and Table B.13.

NOTE 1 For VT version 5 and later, the ACK key sends messages (pressed, held, released) consistent with Key and Button objects. In VT version 4 and prior, this was not well defined and led to variations in implementation.

If a Key object or non latchable Button object is erased from the screen (e.g. due to a Change Active Mask command, Change Soft Key Mask command, Hide/Show Object command, etc) while it is activated, the VT shall send a Soft Key Activation message or Button Activation message indicating released to the erased object on its parent Data Mask. The VT shall then ignore the physical key until the operator has released it. If a Button object is moved to a new location on the active mask, while it is still held, the behaviour depends on the method of activation. If the Button object was activated by a physical key, then the Button object stays pressed. If the Button object was activated by touch screen, pointing device or similar, and the object is moved to a location that is no longer under the touch point, the VT shall send a Button Activation message indicating released and shall then ignore the touch until the operator physically releases. On touchscreen VTs, if the operator presses, then slides his/her finger or pointing device off of the Button or Key object, and if the VT is capable of tracking the operator's finger or pointing device, the VT shall send a Soft Key Activation message or Button Activation message indicating abort. If a pressed Button object is moved such that it is no longer under the touch point while the operator is still pressing or holding the button, AND the operator also slides his/her finger or pointing device off of the Button object, whichever event occurs first shall take priority and the prior described rules shall apply. If the operator is able to track the move, and continue to hold a Button object while it is being moved, the rules above still apply and if the Button moves away from the original touch point, the VT shall auto generate a release event as described above.

For example, when changing the visible Data Mask, the VT shall send the Soft Key Activation message indicating released for the activated object for the previous mask. It shall not send a Soft Key Activation message for the new mask.

Soft Key Masks are considered to be child objects of the Data/Alarm Masks. If the Active Data/Alarm Mask is changed, then the Soft Key Mask is also changed — even if the new Data/Alarm Mask happens to use the same Soft Key Mask.

For example, if a Soft Key is pressed and the Data/Alarm Mask is changed to a new Data/Alarm Mask that is using the same Soft Key Mask, the VT shall send a Soft Key Activation message indicating released and shall then ignore the Soft Key until it has been released. The VT shall also execute the On Hide and On Show macros for the Soft Key Mask.

If a Change Active Mask command or a Change Soft Key Mask command results in no change of the current Active Data/Alarm Mask or Active Soft Key Mask, then there is also no change to the pressed/held/released state of any Buttons or Soft Keys, and the On Hide and On Show macros are not executed either.

NOTE 2 In VT version 5 and prior, this was not well defined and led to variations in implementation.

When the "VT supports simultaneous activation of all combinations of Physical Soft Keys" bit (see D.9) is zero, the VT shall only support the activation of a single Soft Key at a time and only in the prescribed sequence of <no Keys pressed>:<Key pressed>: [<Key held>]:<Key released>:<no Keys pressed>. If a second Soft Key is pressed while a first Soft Key has already been detected as pressed/held, it shall be ignored. When simultaneous activation is supported, the VT sends overlapping messages (e.g. <no Keys pressed>:<Key 1 pressed>: [<Key 1 held>]:<Key 2 pressed>:<Key 1 released>:<Key 2 released>:<no Keys pressed>).

When the "VT supports simultaneous activation of all combinations of Buttons" bit (see D.9) is zero, the VT shall only support the activation of a single Button at a time and only in the prescribed sequence of <no Buttons pressed>:<Button pressed>: [<Button held>]:<Button released>:<no Button pressed>. If a second Button is pressed while a first Button has already been detected as pressed/held, it shall be ignored. When simultaneous activation is supported, the VT sends overlapping messages (e.g. <no Buttons pressed>:<Button 1 pressed>: [<Button 1 held>]:<Button 2 pressed>:<Button 1 released>:<Button 2 released>:<no Buttons pressed>).

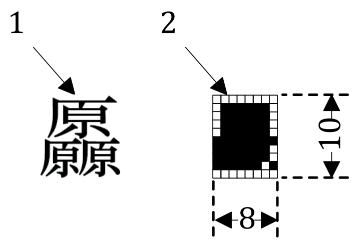
If a Key or Button is found to be pressed at power on, it shall not be reported as held; however this can be cause for VT to report a diagnostic message to the operator.

4.6.19 Font rendering

4.6.19.1 General

Annex K contains the required, and optional, character sets that the VT shall support. Presentation requirements are defined in this section.

The working set design should consider that the character set that can be accessible in version 4 and later VTs can include characters, symbols, and ideograms that cannot render legibly for all font sizes (see Figure 16). In cases where the working set design needs to ensure the presentation at a lower resolution, it can be necessary to deliver the information in a different language, encoding, or with a picture graphic object.



Key

1 high-resolution ideogram

2 illegible representation of high resolution

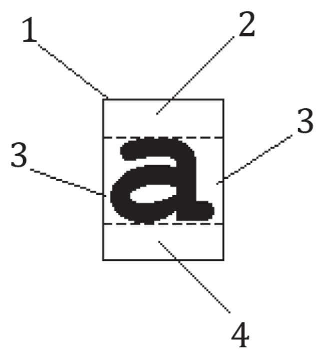
Figure 16 — Font Detail compared to Resolution

The precise definition of the text presentation, for WideString encoded text, is beyond the scope of this document, and will typically be employed by an advanced layout and/or font rendering engine. UNICODE has many “control codes” defined that can affect the presentation of the text. As a result, the text layout is deferred to the UNICODE layout engine and/or font rendering engine and might not precisely match the requirements defined herein. It is expected that justification will be preserved, and it is recognized that auto-wrap and non-printing character rules might not match those defined in the following sections. In the case of a conflict between embedded control codes and the attributes described below, the UNICODE control codes shall take precedence if supported. Control codes that are not supported should not take up any space in the presentation. Some rendering engines allow justification to be set before WideString text is rendered.

4.6.19.2 Text justification

Text based objects have a justification attribute. Field justification indicates how a text string is positioned horizontally and vertically within the field defined by the width and height attributes. In version 3 and prior VTs, text justification was done on a character basis but was not precisely defined. In version 4 and later VTs, text justification is always done on a graphical (i.e. pixel) basis.

The extents of a text character (see Figure 17) shall be graphically justified as described in the following sections but some white space is permissible if the VT’s font rendering engine reserves space for ascenders and descenders or for the character itself.



Key

1	graphical extents of the character
2	ascender area can create white space depending on VT design
3	character rendering itself can create white space depending on VT design
4	descender area can create white space depending on VT design

Figure 17 — Graphical Extents of a Character

During data input of a text based object, the VT designer can choose to suppress justification until the field is closed after data input.

NOTE These modifications are done for visual justification — the stored value is not modified.

Table 6 — Space handling based on justification

	No auto-wrap			Auto-wrap		
	Left justification	**Middle justification**	**Right justification**	**Left justification**	**Middle justification**	**Right justification**
Spaces at the beginning of the first line of a text	n	r	+	n	r	n
Spaces at the end of the last line of a text	+	r	r	+	r	r
Spaces at the beginning of a line after an auto-wrap	–	–	–	r	r	n
Spaces at the end of a line before an auto-wrap	–	–	–	+	r	r

Table 6 *(continued)*

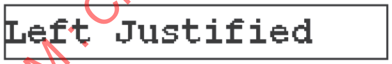
	No auto-wrap			Auto-wrap		
	Left justification	**Middle justification**	**Right justification**	**Left justification**	**Middle justification**	**Right justification**
Spaces at the beginning of a line after a forced line break	n	r	+	n	r	n
Spaces at the end of a line before a forced line break	+	r	r	+	r	r
<Space> handling legend: n = spaces are not removed r = spaces are removed − = spaces do not exist + = spaces do not have an effect on layout NOTE The NBSP character (0xA0) provides a "non-breaking space" and behaves as other non-space characters.						

4.6.19.2.1 Horizontal left justification

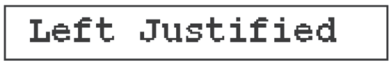
When left justified, the VT shall not remove any leading spaces and the first character in the string is positioned and visible at the left side of the text field area. If auto-wrapping is enabled, the rules of auto-wrapping overrule and leading spaces on subsequent lines are trimmed before justification. If the string does not fit in the defined text field area, and auto-wrapping is not enabled, the string shall be graphically clipped on the right side. If auto-wrapping, justification rules apply to each new line. Blank lines are not removed.

In the examples below, the box represents the extents of the defined text field area.

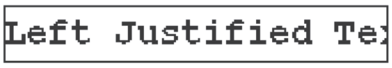
EXAMPLE 1 Left justification, no auto-wrap, no leading spaces, "Left Justified":



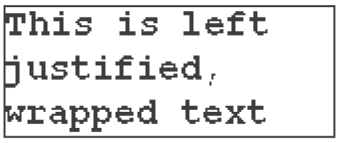
EXAMPLE 2 Left justification, no auto-wrap, one leading space, "Left Justified":



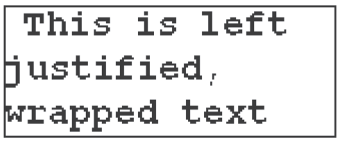
EXAMPLE 3 Left justification, no auto-wrap, clipping on the right, "Left Justified Text":



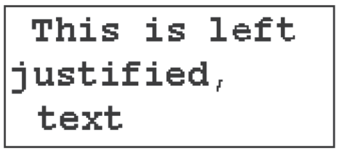
EXAMPLE 4 Left justification, auto-wrap, no leading spaces:



EXAMPLE 5 Left justification, auto-wrap, one leading space:



EXAMPLE 6 Left justification, no auto-wrap, one leading space, one space before and after second <CR> and one trailing space (no space is removed, the space before the <CR> and the trailing space have no effect), “This is left<CR>justified, <CR> text”:



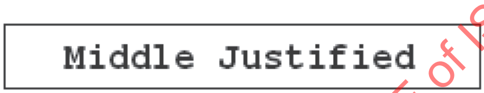
This is left
justified,
text

4.6.19.2.2 Horizontal middle justification

When middle justified, the VT shall remove all leading and trailing spaces before justifying the string. The string shall be centred in the text field on a pixel basis. If the string does not fit in the defined text field area, and auto-wrapping is not enabled, the string shall be graphically clipped on the left and right sides. If auto-wrapping, justification rules apply to each new line. Blank lines are not removed.

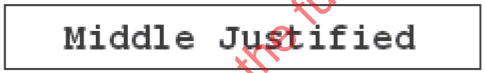
In the examples below, the box represents the extents of the defined text field area.

EXAMPLE 1 Middle justification, no auto-wrap, no leading or trailing spaces, “Middle Justified”:



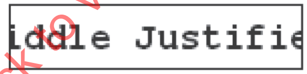
Middle Justified

EXAMPLE 2 Middle justification, no auto-wrap, five leading and one trailing space (leading and trailing spaces are removed), “Middle Justified”:



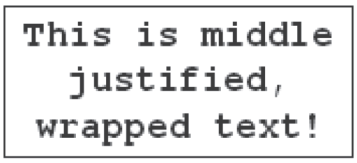
Middle Justified

EXAMPLE 3 Middle justification, no auto-wrap, clipping on the left and right, “Middle Justified”:



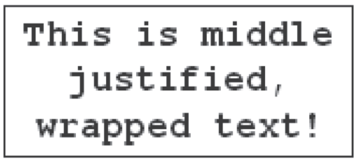
iddle Justifie

EXAMPLE 4 Middle justification, auto-wrap, no leading or trailing spaces, “This is middle justified, wrapped text!”:



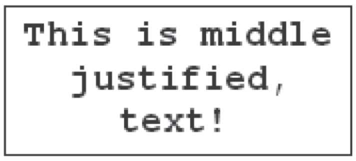
This is middle
justified,
wrapped text!

EXAMPLE 5 Middle justification, auto-wrap, one leading and one trailing space (leading and trailing spaces are removed), “This is middle justified, wrapped text!”:



This is middle
justified,
wrapped text!

EXAMPLE 6 Middle justification, no auto-wrap, one leading space, one space before and after the second <CR> and one trailing space (all spaces are removed), “This is middle<CR>justified, <CR> text!”:



This is middle
justified,
text!

4.6.19.2.3 Horizontal right justification

When right justified, the VT shall remove any trailing spaces before justification and the last character in the string is positioned and visible at the right side of the text field area. If the string does not fit in the defined text field area, and auto-wrapping is not enabled, the string shall be graphically clipped on the left side. If auto-wrapping, justification rules apply to each new line. Blank lines are not removed.

In the examples below, the box represents the extents of the defined text field area.

EXAMPLE 1 Right justification, no auto-wrap, no trailing spaces, “Right Justified”:

```
  Right Justified
```

EXAMPLE 2 Right justification, no auto-wrap, one trailing space (space is removed), “Right Justified”:

```
  Right Justified
```

EXAMPLE 3 Right justification, no auto-wrap, clipping on the left, “Right Justified Text”:

```
ght Justified Text
```

EXAMPLE 4 Right justification, auto-wrap, no trailing spaces, “This is right justified, wrapped text”:

```
 This is right
   justified,
  wrapped text
```

EXAMPLE 5 Right justification, auto-wrap, one trailing space (space is removed), “This is right justified, wrapped text”:

```
 This is right
   justified,
  wrapped text
```

EXAMPLE 6 Right justification, no auto-wrap, one space before and after second <CR> and one trailing space (space before the <CR> and trailing space are removed, the space after the <CR> has no effect), “This is right<CR>justified, <CR> text”:

```
 This is right
   justified,
          text
```

4.6.19.2.4 Vertical top justification

Vertical top justification is available in VT version 4 and later. When vertical top justification is enabled, the VT shall display the text string starting at the extreme top of the defined text field area. This rule applies regardless of the auto-wrapping attribute. Blank lines are not removed.

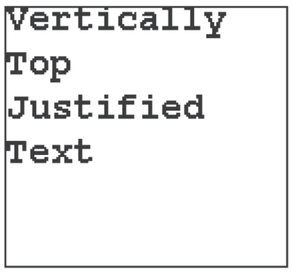
In the examples below, the box represents the extents of the defined text field area and in all examples the text is also left justified.

EXAMPLE 1 Top justification, no auto-wrap, “Vertical Top”:

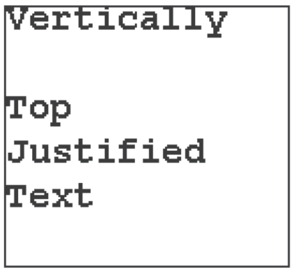
```
Vertical Top

```

EXAMPLE 2 Top justification, auto-wrap, “Vertically Top Justified Text”:



EXAMPLE 3 Top justification, auto-wrap, “Vertically<CR><CR>Top Justified Text”:

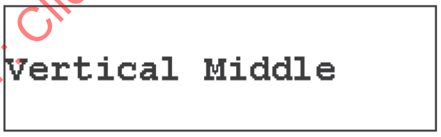


4.6.19.2.5 Vertical middle justification

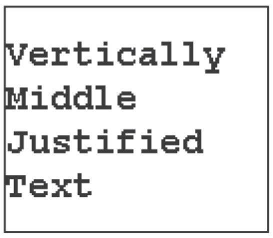
Vertical middle justification is available in VT version 4 and later. When vertical middle justification is enabled, the VT shall display the text string graphically centred vertically in the defined text field area. This rule applies regardless of the auto-wrapping attribute. Blank lines are not removed.

In the examples below, the box represents the extents of the defined text field area and in all examples the text is also left justified.

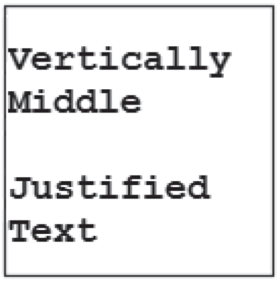
EXAMPLE 1 Vertical middle justification, no auto-wrap, “Vertical Middle”:



EXAMPLE 2 Vertically middle justification, auto-wrap, “Vertically Middle Justified Text”:



EXAMPLE 3 Vertically middle justification, auto-wrap, “Vertically Middle<CR><CR>Justified Text”:

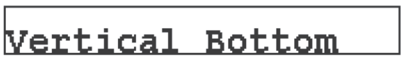


4.6.19.2.6 Vertical bottom justification

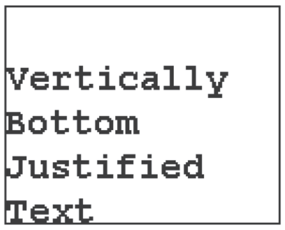
Vertical bottom justification is available in VT version 4 and later. When vertical bottom justification is enabled, the VT shall display the text string with the bottom edge of the text block along the bottom edge of the defined text field area. This rule applies regardless of the auto-wrapping attribute. Blank lines are not removed.

In the examples below, the box represents the extents of the defined text field area and in all examples the text is also left justified.

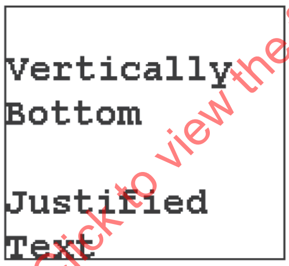
EXAMPLE 1 Bottom justification, no auto-wrap, "Vertical Bottom":



EXAMPLE 2 Bottom justification, auto-wrap, "Vertically Bottom Justified Text":



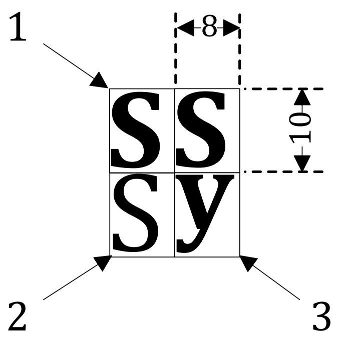
EXAMPLE 3 Bottom justification, auto-wrap, "Vertically Bottom<CR><CR>Justified Text":



4.6.19.3 Non-proportional fonts

The VT shall support non-proportional block fonts. Sizes are always given in X-Y pairs. For example, 8 × 10 indicates a character size of 8 pixels wide by 10 pixels high. Characters shall not exceed the size of the box defined by the font size, regardless of style. For example, an 8 × 10 font set to bold and italics shall still fit inside the 8 × 10 pixel area. It is suggested that space be left on the bottom and right sides on the inside of the box to accommodate characters placed side by side or row by row (see Figure 18).

Dimensions in pixels



Key

1	bold
2	normal (upright)
3	bold italic

Figure 18 — 8 × 10 fonts — Example

The VT designer can choose the font sizes and styles to be made available but the default 6 × 8 font, normal (upright) style, is a minimum requirement. The Get Text Font Data message can be used by a Working Set to determine the VT's font capabilities. Characters can be rendered transparent (background shows through) or opaque (solid background colour) depending on the options attribute of the applicable object.

4.6.19.4 Proportional fonts

a) In version 4 and later VTs, as an option, the VT design can allow for proportional font rendering. In this case the width of each character is variable and the height attribute is fully scalable in the range from 8 pixels up to and including the largest supported font height as identified in D.7 Get Text Font Data response. As a result, only the height attribute of the font size applies and the width attribute is ignored. Therefore, the rendered characters shall not exceed the height of the chosen font size.

b) For example, if the VT responds to a Get Text Font Data message with Byte 6 = 10_{16}, and Byte 7 = 02_{16}, then the largest reported non-proportional font size is 48 pixels wide by 64 pixels in height. If the VT has indicated support for proportional font rendering (Byte 8 bit 7 = 1), then the VT supports a proportional font height from 8 up to and including the value 64, in 1 pixel resolution. Due to characteristics of the implemented font rendering engine, and the shape of the characters, 1 pixel resolution might not be detectable for every character, even though it is supported.

c) Normal clipping rules still apply and Working Set designers have to be aware of the variable nature of the font width and therefore assign sufficient width to the text field. Implements can query the VT's font capabilities, including the support of proportional font rendering, via the Get Text Font Data message and can choose the proportional rendering option in the font style attribute of the Font Attributes object. If the implement requests a proportional font rendering and the VT does not support this option, the VT shall respond in one of two ways:

— if a Font Attributes object indicates proportional font during the validation of an object pool, the object pool shall be rejected;

— if a Change Font Attributes command is received with invalid size and/or font type, a Change Font Attributes response shall be sent indicating an error in the size and/or type.

The following general rules apply for the support of Proportional font rendering:

— a VT that indicates support for proportional fonts shall support the full height scaling range of 8 to the largest supported font size as identified in Get Text Font Data response. Character width is recognized to be variable.

Before using proportional fonts, Working Sets shall query the VT to determine if proportional font rendering is supported by the VT. If not supported, only non-proportional fonts shall be used in the pool upload and subsequent Change Font Attributes commands.

4.6.19.5 Auto-wrap

If the amount of text to display is longer than the width of the text object, and auto-wrap is enabled, then regardless of non-proportional or proportional rendering, the VT shall format and wrap the text into the next line(s). The text shall not exceed the boundaries defined by the width of the text object. Wrapping shall occur under these conditions:

— Space (20_{16}) between words.

— Soft Hyphen (AD_{16}). When wrapping occurs on the soft hyphen, the soft hyphen shall be shown before the line break; otherwise the soft hyphen is not shown.

— Hyphen ($2D_{16}$). Wrapping can occur between a hyphen ($2D_{16}$) and the following character, when the Wrap on Hyphen option bit in text objects is TRUE.

— If, after applying the above rules, there is no breaking point in the line, then wrap on the last wholly visible character in the line (see Figure 15).

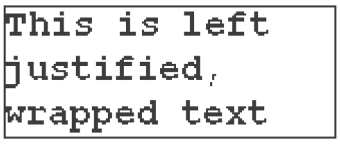
— At line end (see 4.6.19.6).

Leading space characters on the next line shall be suppressed and not considered in additional auto-wrap decisions.

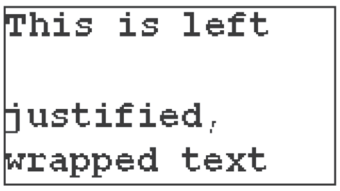
If the auto-wrapping occurs at the position of a "forced" wrap like <CR> both together will interpreted as one line end (Example 1), otherwise they will be handled separately (Example 2).

See also 4.6.19.6.

EXAMPLE 1 Left justification, auto-wrap, some spaces before <CR>, auto-wrapping and <CR> at same position, "This is left <auto-wrapping here><CR>justified, <auto-wrapping here>wrapped text":


```
This is left
justified,
wrapped text
```

EXAMPLE 2 Left justification, auto-wrap, some spaces before <CR>, auto-wrapping one space before <CR>, "This is left <auto-wrapping here> <CR>justified, <auto-wrapping here>wrapped text":


```
This is left

justified,
wrapped text
```

4.6.19.6 Non-printing characters in strings

VT version 3 and prior allowed CR, LF, and BS control characters in strings. The rendered presentation was not precisely defined.

The following clarification applies to VT version 4 and later.

— BS (Back Space) is ignored as are other characters denoted with scissors in Annex K.

— Single CR (Carriage Return) is interpreted as a line end.

— Single LF (Line Feed) is interpreted as a line end.

— Sequence CRLF is interpreted as a line end.

— 00_{16}, or 0000_{16} (WideChar) shall terminate the presentation even if this occurs before the defined string length. All characters following the terminating zero shall be ignored, both for data input and presentation. Editing can increase the apparent string length up to the defined string length, but shall not limit the editing of strings, which is controlled by the length attribute.

Based on these definitions:

— Sequence LFCR is interpreted as two line ends.

Other non-displayable characters shall not advance the cursor during drawing or alignment decision making. The VT can remove these characters from the string to facilitate more efficient processing.

String Data	VT Presentation		
	Left Alignment	Middle Alignment	Right Alignment
String 1: ABCDEF**CR**GHIJKL or ABCDEF**LF**GHIJKL or ABCDEF**CRLF**GHIJKL	ABCDEF GHIJKL	ABCDEF GHIJKL	ABCDEF GHIJKL
String 4: MNOPQR**LFCR**STUVWX	MNOPQR ¶ STUVWX	MNOPQR ¶ STUVWX	MNOPQR ¶ STUVWX
NOTE “¶” shown for examples and would not be visible on the VT			

Figure 19 — CR and LF application to test strings

4.6.19.7 String encoding

Text strings can be encoded with either 8-bit characters (chars) or, starting with VT version 4, with Unicode/ISO10646 characters (WideChars).

VT Version 3 and prior supported Font types are shown in Table K.2 and Table K.3.

VT Version 4 and later supported Font types are shown in Table K.2 through Table K.7 and Table K.8.

The character set is indicated by the Font type attribute of the Font Attributes object.

WideStrings shall be encoded according to UTF-16 (Unicode Transformation Format—16 bit) and therefore they always start with the Byte Order Mark (BOM) character $FEFF_{16}$. BOM is not a displayable character and is not considered part of the text string.

UTF-16 allows both big- and little-endian encoding, but WideStrings in ISO11783-6 shall always be encoded as little-endian.

BOM is used to distinguish between 8-bit strings and WideStrings.

If the first two bytes are FF$_{16}$, FE$_{16}$ it is a WideString, otherwise it is an 8-bit string.

The length attribute of an object or message always indicates the number of bytes in the text string, therefore the number of characters in a WideString shall be found as:

number of WideChar = length/2 — (number of surrogate pairs) — 1

e.g. FF, FE, 41, 00, 42, 00, 3D, D8, 52, DD, 3D, D8, 54, DD=> "AB🕒 🕔", [length = 14, number of surrogate pairs = 2, number of WideChar = 4]

If the length attribute does not indicate an even number of bytes the last byte is ignored.

The VT can support any character defined by Unicode/ISO10646, but as a minimum it shall support the characters listed in Table K.8.

The Font type attribute of the Font Attributes object is ignored for WideStrings.

The VT shall display WideStrings even if the WideStrings contain characters which are not supported by the VT. The VT shall substitute unsupported characters by a displayable character. The displayable character can be VT proprietary (e.g. "☐").

The encoding of Input String object values shall not be changed by the VT, i.e. if an Input String object contains (or references) a WideString, the VT Change String Value message sent by the VT shall also contain a WideString.

Characters above FFFF$_{16}$ are represented by a 32 bit "surrogate pair", which consists of a high surrogate followed by a low surrogate.

The surrogate pair is constructed as follows:

a) S = character—10000$_{16}$;

b) High surrogate = D800$_{16}$ + (S shifted 10 bits to the right);

c) Low surrogate = DC00$_{16}$ + (10 least significant bits of S).

The highest character defined by Unicode and ISO 10646 is 10FFFF$_{16}$, corresponding to the surrogate pair DBFF$_{16}$, DFFF$_{16}$.

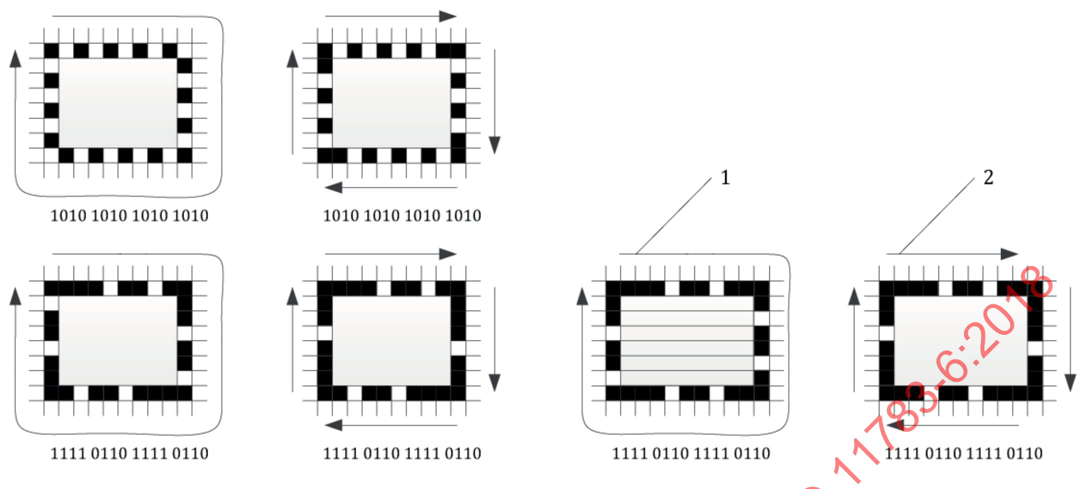
4.6.20 Object Rendering Accuracy, Quality and VT Developer Freedom

It is in the intent of this document to enable interoperability among equipment from different manufacturers, and to do so in a manner that successfully conveys the original design accurately enough for proper interpretation by the operator.

Many of the VT objects in this document do not have all elements of their presentation explicitly defined. The implementation chosen by the developer can be one that favours computing performance, visual style, or other factors that are outside the definitions in this document. The Output Meter object is an example where the developer defines some elements of the presentation. For example, even though the bounding width, height, and values are clearly defined, the developer can define the size and shape of the needle. Other examples include the Output Linear Bar Graph object (e.g. fill, set point mark, tic mark size and position), and Output Line objects not conforming to a strict 0, 45, or 90-degree orientation. Drawing algorithms as can be used to render these types of objects can produce similar but varying results (e.g. two diagonal lines drawn with slightly different line drawing algorithms). Pixel level accuracy in this case might not be possible. Where pixel level accuracy is required, the designer should consider a Picture Graphic object, while also ensuring that the VT does not scale this object (e.g. the Width attribute is equal to the Actual width attribute).

4.6.21 Line art and filling output shape objects

The VT can render the line-art using either the continuous line art method or the restarting line art method (see Figure 20). The line art starting point should be consistently applied (e.g. first point in a polygon, upper-left for a rectangle, starting angle for an ellipse). The exact presentation of the line art, especially for thicker lines and in combination with line suppression is determined by the VT design.

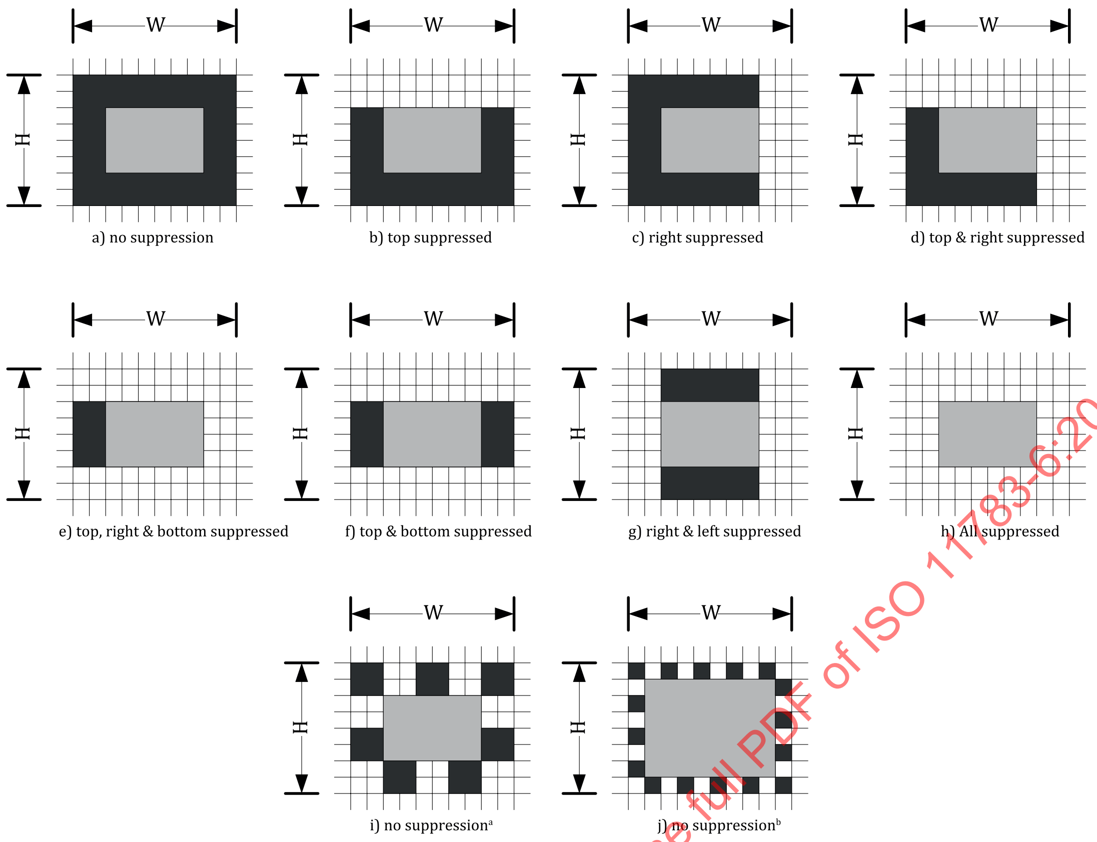


Key

1	indicates application of continuous line art pattern
2	indicates application of restarting line art pattern

Figure 20 — Continuous and restarting line art pattern examples

When solid-filling output shape objects on the VT, flood-fill or boundary-fill type algorithms are not suitable, since objects can be overlaid and interrupt the fill. There are also performance problems with this type of approach. Scan-line type fills shall be implemented. In addition, only the interior area of the object, not including any pixel that is/would be part of the border, shall be included in the fill. Incorrect filling would be particularly visible when line art is used on the border or when the border is completely or partially suppressed (i.e. rectangles).



a Rectangle width = 10, height = 8, line width = 2, filltype = solid grey, line art 1010.

b Rectangle width = 10, height = 8, line width = 1, filltype = solid grey, line art 1010.

NOTE All rectangles width = 10, height = 8, line width = 2, filltype = solid grey.

Figure 21 — Rectangle line suppression and filling examples

Pattern fills of Output Shape objects shall be done according to the following rules:

— The pattern shall be a Picture Graphic object whose width matches the restrictions as specified in Table B.50. The raw data are used for the pattern and the object is not scaled regardless of the attributes of the object.

— The upper left corner of the pattern buffer is anchored to the upper left corner of the VT's physical Data Mask, individual designator, or user-layout window mask and repeats across and also down. This rule ensures that the pattern matches between objects in the Data Mask Area and that the pattern fill looks the same on all VT designs.

— Transparency and flashing option attributes shall be ignored for fill patterns.

Filling and line suppression examples are shown in Figure 21 to Figure 23.

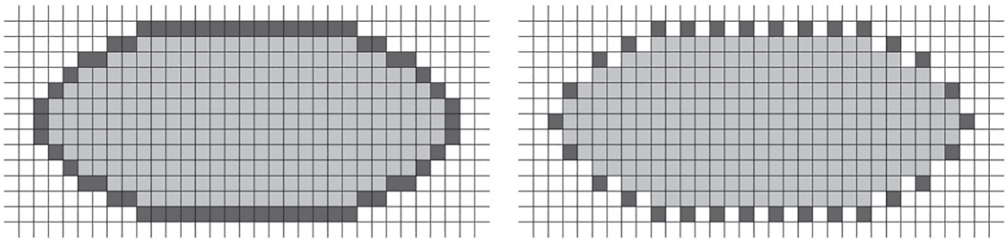


Figure 22 — Ellipse filling examples (Without and with border line art)

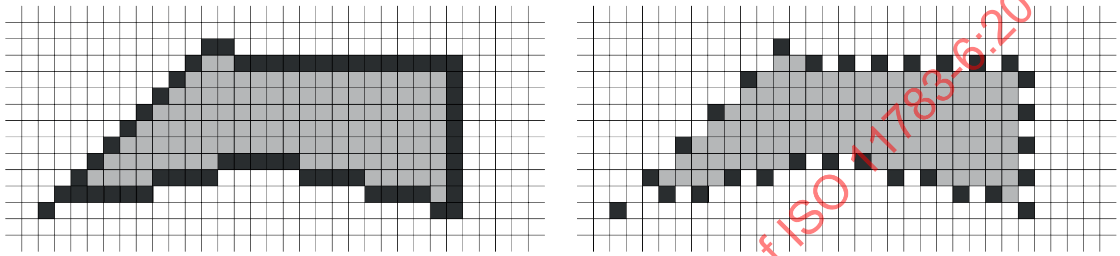


Figure 23 — Polygon filling examples (Without and with border line art)

4.6.22 Events

4.6.22.1 General

Manipulation of objects in an object pool by a Working Set or by the VT causes certain events to occur. Events can be caused by commands or by VT actions in response to a command or by other events. Many objects defined by this part of ISO 11783 have an optional list of event and Macro groupings.

If the occurring event has one or more Macros associated with it, the Macro or set of Macros is executed by the VT when the command has been accepted by the VT as a valid command. The Macro shall be executed even if the command does not result in the update of an object parameter, value, attribute or display. Macros are executed in the order they are encountered in the event/Macro list. Using events and Macros can make the VT operator interface more responsive, since the Working Set Master does not need to be directly involved in responding to the event.

EXAMPLE 1 A Soft Key press event could cause an appropriate Data Mask to be made active.

EXAMPLE 2 A Soft Key press event could cause two separate Macros to execute, one to make a new Data Mask active and one to control the audio signal.

EXAMPLE 3 A Change Numeric Value command with an invalid value causes rejection of the command and an associated Macro is not executed. A Change Numeric Value command with a valid value, even if the value is the same as already in the value field, will cause the associated Macro to execute.

EXAMPLE 4 A container with an On Hide Macro is already hidden. A Hide Show command with the parameter set to hide the container is valid and will cause the associated Macro to execute.

NOTE The same event id can be listed more than once in the event/Macro list of any given object.

4.6.22.2 Macro references — VT version 4 and prior

Version 4 and prior VTs provide 2 bytes for each event and Macro grouping within the object definition. This means one byte for the Event ID and one byte for the Macro ID, limiting the Object IDs for Macro objects to the range of 0 to 255.

A VT version 4 single Macro reference has the following structure:

— First byte: Event ID (in the range of 0 to 254_{10})

— Second byte: Macro ID (in the range of 0 to 255_{10})

4.6.22.3 Macro references — VT version 5 and later

Version 5 and later VTs, in addition to the 8-bit Macro Object IDs, shall support Macros with an Object ID in the range of 0 to 65534, requiring the use of 16-bit Object IDs for Macros. To maintain backwards compatibility, Macro references within objects are based on the same structure with 2 bytes per grouping, but two of these groupings are used to reference a macro with 16-bit Object ID. An Event ID of 255 in the first byte of the Macro reference indicates that two groupings shall be concatenated to a single grouping with a 16-bit Macro Object ID reference.

A VT version 5 single 8-bit Macro reference has the following structure:

— First byte: Event ID (in the range of 0 to 254_{10})

— Second byte: Macro ID (in the range of 0 to 255_{10})

EXAMPLE 1 A Container object references a Macro with 8-bit Object ID 7_{10} (07_{16}) that shall be executed when the container is hidden. The event and Macro grouping within the definition of the Container object is as follows:

First byte: Event ID = 04_{16} (on hide)

Second byte: Macro ID = 07_{16}

A VT version 5 single 16-bit Macro reference has the following structure:

— First byte: Event ID 255_{10} (Use Extended Macro Reference)

— Second byte: Low byte of Macro ID (Macro ID in the range of 0 to 65534_{10})

— Third byte: Event ID (in the range of 0 to 254_{10})

— Fourth byte: High byte of Macro ID (Macro ID in the range of 0 to 65534_{10})

EXAMPLE 2 A Container object references a Macro with 16-bit Object ID 7000_{10} ($1B58_{16}$) that shall be executed when the container is hidden. The event and Macro grouping within the definition of the Container object is as follows:

First byte: Event ID = FF_{16} (Use Extended Macro Reference)

Second byte: Macro ID = 58_{16}

Third byte: Event ID = 04_{16} (on hide)

Fourth byte: Macro ID = $1B_{16}$

4.6.23 Touch screens and pointing devices

The VT design can optionally support a touch screen or a pointing method such as a mouse or joystick. The Working Set can determine the VT's capabilities in this regard by using a Get Hardware message and then make necessary adjustments to its object pool. A Button object is defined to allow touchable or clickable buttons to be included in a Data Mask. A Pointing Event message is defined to allow the VT to notify the active Working Set that an area of the Data Mask or Free Form Window Mask (type 0) *not associated* with a button or other input object has been touched or clicked on. VT version 4 and 5 included Alarm Mask, which has been deprecated with VT version 6 and later.

4.6.24 Proprietary Means

There are various proprietary means supported by the VT (Proprietary Objects, Proprietary Events, Proprietary Colours, Proprietary Commands, and Proprietary Fonts). Use of these proprietary means where the Working Set is provided by a different manufacturer than that which provides the VT is not recommended in order to provide maximum ISO compatibility. As these items are proprietary, the Working Set or VT manufacturer can change their proprietary means without disclosure.

Further, and in the context of the Proprietary Objects, it is not possible to parse an object for which the definition is not known, and attempting to do so can cause the VT to reject the pool.

Additionally, Working Sets and VTs can exchange, and support, higher version messages and objects without violating this standard as long as those features do not contradict the requirements defined for the lower version (e.g. a version 2 VT can support Font type 5 — Cyrillic, even though not part of the standard until version 4). Even as this mechanism leverages the standard, it shall be considered a proprietary means as it is not defined in the lower version. Therefore, a proprietary means is required to determine if the connected VT or Working Set supports the newer features, and the Working Set should not assume that the new features are supported to avoid undefined behaviour.

To support the transition where older Working Sets want to use capabilities from higher VT versions, it is recognized that the Working Set can deliver features defined for a higher version VT in its object pool. The VT shall reject the pool when there are unsupported features in the pool. The VT can accept the pool when all features are fully supported in both presentation and for any specific messaging behaviours.

4.6.25 VT Number

By default, VT’s shall be factory set to function instance zero (0), but will retain the function instance as configured by the operator. A mechanism is required in order to conveniently resolve conflicts in the case where there are multiple VT’s with the same function instance and in the case where there is no VT with function instance zero. The VT shall be responsible for providing a proprietary means for setting the function instance from the display itself. This means shall ensure that duplicate function instances between VT’s are not created. The new function instance shall not be used until a re-initialization of the VT is performed (see 4.6.6). The VT with function instance zero (0) is defined as the “primary VT”.

The proprietary means to set the function instance shall represent to the operator a VT Number (see Clause 3). In this way, VTs from all manufacturers will present a consistent numbering scheme for the operator to choose the primary (and secondary) VT(s). To facilitate easy VT identification, the Identify VT message (see D.18) can be used to cause all VTs to show their current setting.

4.6.26 Packet Padding

All VT to ECU and ECU to VT messages that are not explicitly defined to contain exactly 8 data bytes shall be padded to the 8 byte boundary with FF_{16}.

4.6.27 Momentary or Non-latched Means

A Momentary or Non-latched Means will be ON, while activated by the operator, and will automatically return to OFF when the operator stops activating it. Examples of Momentary Means includes, but is not limited to, Buttons, Keys, Touch Events and Auxiliary Input controls.

When a Momentary Means is activated, it shall report its transition from OFF to ON.

If a Momentary Means stays activated, it shall report its status as Held. The Held status shall be repeated until the Momentary Means is no longer activated. Specific values for the timing are given in the description of each Momentary means.

When a Momentary Means is no longer activated, it shall report its transition from ON/Held to OFF.

Note that ON, Held and OFF are generic terms, the descriptions of individual Momentary Means can use other terms, e.g. for buttons the terms are Pressed, Held and Released.

4.6.28 Unsupported Objects

VT version 4 and later defines the capability for optional support of some objects. The Working Set can query the VT to determine which objects are supported with the Get Supported Objects message (see D.14). Even with this optionality, there are requirements imposed on the VT as follows:

— The Working Set can transfer an object pool containing any non-proprietary object that is defined compatible with the VT version reported in Get Memory response.

— Non-proprietary objects that are not listed as supported by the VT shall still be parsed, but they shall not be functionally supported by the VT. In this way, some Working Sets can choose to use the same object pool in both cases.

— The Working Set should not send commands directed to an object that is unsupported by the target VT.

— If the Working Set sends a command directed to an object that is unsupported by the VT, the VT will send the corresponding response with a non-zero Error Code.

— The specific value of the Error Code is dependent on the design of the VT.

EXAMPLE Example: A Working Set has an object pool that contains a Graphics Context object. This pool can be delivered to any VT version 4 and later VT, even if the target VT does not list the Graphics Context object as supported. The Working Set then sends a Change Background Colour command targeting the Graphics Context object, The VT sends a Change Background Colour response message with a non-zero error code.

4.6.29 Error codes

In case errors are detected during execution of a command, the response message to that command shall contain at least one applicable error code. In case of multiple errors detected the response message to that command should contain all detected error codes.

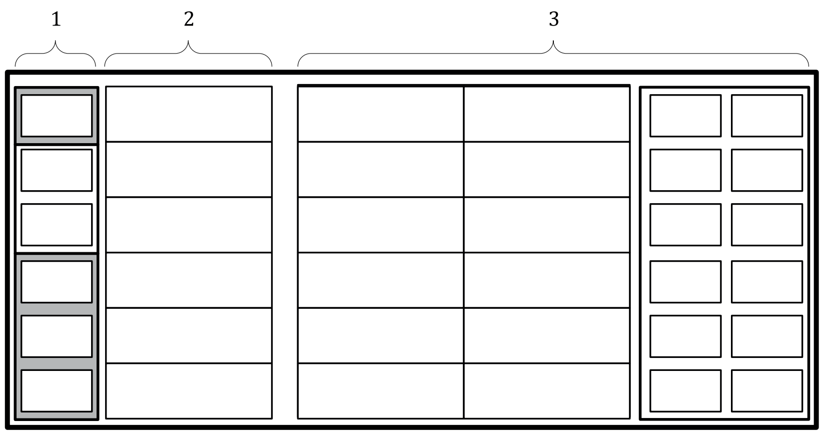
4.7 Displaying Data from Multiple Working Sets on One Mask

4.7.1 General

4.7 and all subordinate clauses apply to VT version 4 and later.

4.7.1.1 Displaying Data on one screen

The VT can provide a means where data from multiple Working Sets can be made available on one screen. Depending upon the VT design, it is also possible that data from multiple Working Sets can be made available simultaneous to the VTs standard Data Mask and Soft Key Mask. See example in Figure 24.



Key

1	Key Group Objects in non-VT area (shaded regions indicate independent Key Groups)
2	Window Mask Objects in non-VT area
3	VT area (presenting standard VT screen or User-Layout Data Mask and User-Layout Soft Key Mask)

Figure 24 — Displaying data from multiple Working Sets — Example

4.7.1.2 Minimal Requirements

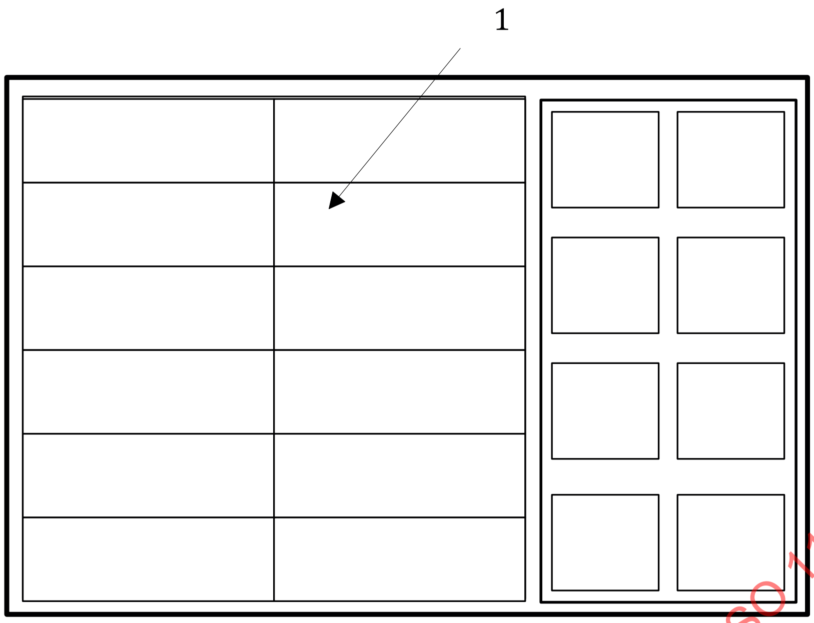
This is an optional feature. However, as a minimum, the VT shall be required to parse the Window Mask and Key Group objects even if not supported. This allows Working Sets to upload these objects to all version 4 and later VTs without modification of the object pool. If this feature is supported, the VT shall support the "free form" (type 0) window mask type of the Window Mask object as a minimum. Any number of the other non-zero window mask types can also be supported as desired. Working Set designs can desire the use of any of the window mask types so implementation, by the VT design, of all window mask types is encouraged. If the Working Set uploads a Window Mask object with an unsupported window mask type, the VT shall parse, but ignore, this object and no errors shall be raised. Unsupported window mask types would not be presented to an operator for selection.

If the Working Set chooses to participate in this feature, it shall upload Window Mask and Key Group objects as part of the object pool. Since the VT shall parse but ignore any Window Mask objects with unsupported window mask type, no modification of the object pool is necessary.

4.7.2 User-Layout Data Mask

The VT can support any number of User-Layout Data Masks. User-Layout Data Masks are special Data Mask objects that are owned by the VT. The VT shall provide a mechanism to allow the operator to access the available User-Layout Data Masks.

Each User-Layout Data Mask is the same size as a standard Data Mask object but is divided into a grid of Window Cells with exactly two columns and six rows (see Figure 25).



Key

1	User-Layout Data Mask showing 12 Window Cells

Figure 25 — User-Layout Data Mask

4.7.3 Window Mask object

Window Mask objects, optionally supplied in the object pools of Working Sets, are placed into a specific User-Layout Data Mask grid as desired by the operator of the VT. An individual Window Mask object can take up more than one Window Cell, and can take up the entire 2 × 6 grid. 1 × 1 Window Masks are recommended where possible to allow the operator to select from a wide range of Window Mask objects to display in each User-Layout Data Mask (see example in Figure 26).

NOTE Window Mask objects are not limited to the sizes shown in Figure 26.

4.7.4 Window Mask content

4.7.4.1 Presentations

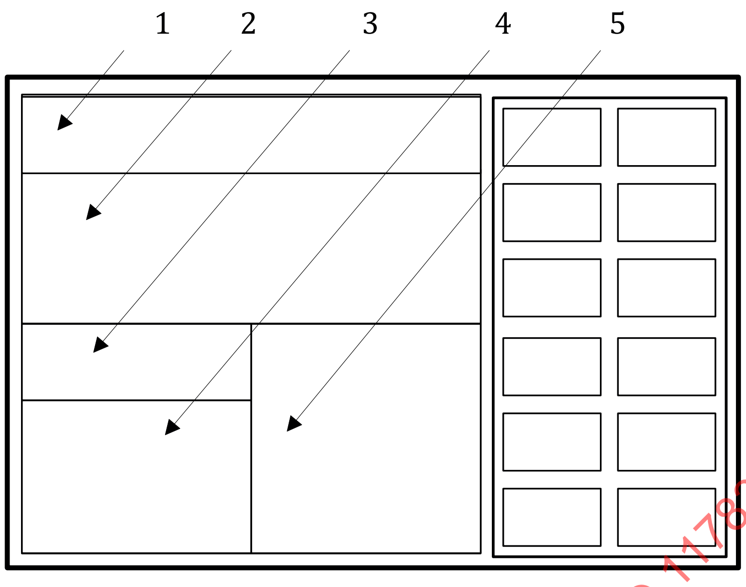
A Window Mask object can display content using two different types of presentations. This is controlled by the Working Set, using a Window Type attribute. More details are available in 4.7.16.

4.7.4.2 Working Set defined presentation

If the Window Mask object has a Window Type of zero, the Window Mask object behaves very similar to a Data Mask object (other than the “size” of the Window Mask object). As with a Data Mask object, all content and presentation is defined by the Working Set.

4.7.4.3 VT defined presentation

If the Window Mask object has a Window Type in the allowed set of non-zero values, the Working Set provides references to a specific set of objects, and the VT then controls the presentation of those objects. The VT is free to ignore any visual formatting attributes in the referenced objects. This capability allows information from different manufacturers to have a consistent look and feel and interaction with the operator when combined into the VT.



Key

1	Window Mask (2 × 1)	4	Window Mask (1 × 2)
2	Window Mask (2 × 2)	5	Window Mask (1 × 3)
3	Window Mask (1 × 1)		

Figure 26 — Window Mask objects — Example

If User-Layout Data Masks are supported, the VT shall provide a proprietary mapping mechanism to allow the operator to select which Window Mask objects are placed in which Window Cells in each of the User-Layout Data Masks. The VT shall prevent the operator from choosing a Window Mask object that does not fit in the selected Window Cell(s). When a Window Cell is selected in the mapping screen by the operator, the VT shall present the list of all Window Mask objects that can fit in the cell(s), provided the Window Mask object is available (see options attribute in the Window Mask object). The VT shall store the operator selected layout of each of the User-Layout Data Masks in non-volatile memory for recall on next power up. If the Working Set that provided the Window Cell is not present, the Window Cell shall be blanked (i.e. filled with the background colour).

If a Window Mask object's options attribute indicates that the mask is not available, the VT shall blank (i.e. fill with the background colour) the associated Window Cell so that the Window Mask is not visible. A change in state of the availability option can occur at runtime.

If the VT is Version 4 or later but does not support User-Layout Data Masks, the Working Set can still include these objects in the object pool transfer. The VT shall parse the Window Mask object and can then discard it.

The Window Mask object width shall be equal to the width of the Window Cell(s) that it occupies. The Window Mask object height shall be equal to the height of the Window Cell(s) that it occupies.

NOTE Working Set designs are encouraged to participate in the User-Layout Data Mask by uploading Window Mask objects since this can be a feature requested by users.

4.7.5 Window Cell Size and Borders

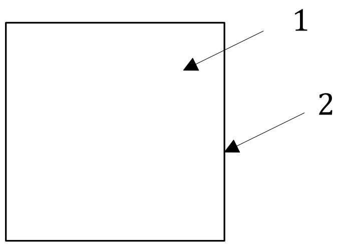
The size of a single Window Cell is based on the 2 × 6 grid in the User-Layout Data Mask. The size of each Window Cell shall be rounded down. Therefore, the size of a Window Cell is defined as follows:

Window Cell Width = Data Mask Width/2 (rounded down)

Window Cell Height = Data Mask Height/6 (rounded down)

The VT can draw a border around each Window Mask object. Whether the borders or any particular part of the borders are actually drawn or not, is proprietary to the VT design. Border drawing is recommended but can be based on an operator choice. The border area shall occupy the outside 1 pixel around the entire Window Mask object and shall be drawn inside the Window Mask object's region. Working Set designs shall be aware of this and therefore it is recommended that no child object be placed on or touching the border area when the free form (type 0) window type is used. See B.19.2 and Figure 27 for an example of the border.

EXAMPLE:



Key

1 Window Mask Region

2 1-pixel border drawn around inside perimeter of Window Mask

Figure 27 — Window Mask Border — Example

The VT shall clip any pixel in the Window Mask object (and its children) that falls outside the Window Mask region.

4.7.6 Window Mask Scaling

Depending on the type of Window Mask, the VT and the Working Set shall cooperate in terms of scaling. If the window type is type 0 (free form), then as with other objects in the object pool, scaling of the Window Mask object and its children is solely the responsibility of the Working Set. Regardless of the resolution of the VT in use, the aspect ratio is known since the User-Layout Data Mask always has a 2 × 6 grid and the Data Mask area is always square. Object pool designers should design the Window Mask object to an appropriate aspect ratio which makes the scaling easier.

EXAMPLE 1 If the object pool is designed for a default 200x200 Data Mask Area, using the equations in 4.7.5, the size of each Window Cell would be:

Window Cell Width = 200/2 = 100 pixels

Window Cell Height = 200/6 = 33 pixels

EXAMPLE 2 Using the Window Cell sizes above when using a VT with a default 200x200 Data Mask Area, a 2x2 Window Mask would use:

Window Mask Width = 100 × 2 = 200 pixels

Window Mask Height = 33 × 2 = 66 pixels

The VT controls most of the layout, formatting and scaling of objects when the Window Mask type is not the Free Form Window Mask (type 0). The exception to this rule is the Window Icon and Button attributes that are pre-scaled by the Working Set. The VT can further scale the Window Icon and Buttons if necessary. More information is contained in the sections that describe the window types greater than type zero (see B.19.2).

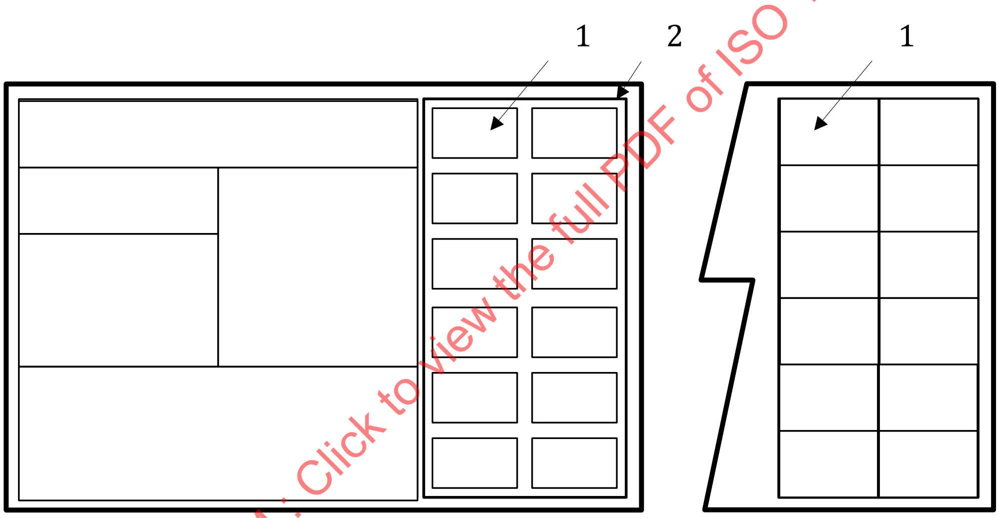
4.7.7 Using Window Masks Outside of User-Layout Data Masks

The VT can optionally use Window Mask objects in Non-VT Screens and Non-VT Areas, if supported by the manufacturer's design (see Figure 24). Soft Keys might or might not be supported when used in non-VT screens and non-VT areas. Working Sets shall be aware that Window Mask object can be used outside of User-Layout Data Mask and therefore shall monitor the VT On User-Layout Hide/Show message and refresh Window Mask objects and keys that are visible regardless of the source address specified in the VT Status message.

4.7.8 User-Layout Soft Key Mask

If the VT supports User-Layout Data Masks, it shall also support one and only one User-Layout Soft Key Mask per User-Layout Data Mask. User-Layout Soft Key Masks are special Soft Key Mask objects that are owned by the VT. Each User-Layout Soft Key Mask is divided into Key Cells (one cell per physical key if Physical Soft Keys are used). The number of Key Cells supported by the User-Layout Data Mask is proprietary to the VT design. Each Key Cell is sized to a normal Soft Key designator. The User-Layout Soft Key Mask is divided into Key Cells as shown in Figure 28.

The VT designer shall decide how many keys per User-Layout Soft Key Mask are supported up to a maximum of 64. If the number of keys supported exceeds the number of physical Soft Keys, the VT shall provide a paging mechanism (identical to the Soft Key Mask object) to allow for proper mapping and operation.



Key

1 Key Cells (same size as Soft Key designator)

2 User-Layout Soft Key Mask

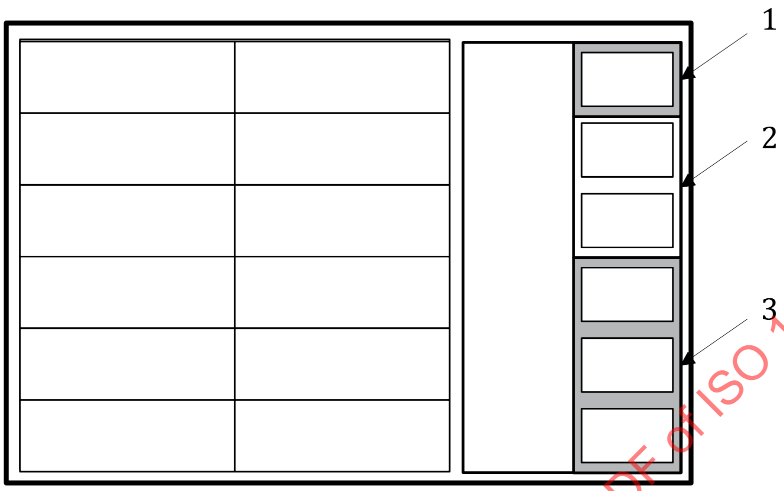
Figure 28 — Key Cell layout — Examples

4.7.9 Key Group Objects

Key Group objects (see B.20), optionally supplied in the object pools of Working Sets, are placed into a User-Layout Soft Key Mask by the operator of the VT. A Key Group object can contain one to four Key objects (although one is typical) and therefore can occupy one or more Key Cells. If User-Layout Data Masks and User-Layout Soft Key Masks are supported, the VT shall provide a proprietary mapping mechanism to allow the operator to select which Key Group objects are placed in which Key Cells in each of the User-Layout Soft Key Masks. When Key Cell(s) are selected in the mapping screen by the operator, the VT shall present the list of all Key Group objects from all Working Sets that provide them, provided the Key Group is available (see options attribute in the Key Group object). The VT shall store the operator selected layout of each of the User-Layout Soft Key Masks, in non-volatile memory, for recall on the next power up. If the Working Set that provided the Key Group Objects is not present, the assigned Key Cell(s) shall be blanked (i.e. filled with the background colour).

If a Key Group object’s options attribute indicates that the Key Group is not available, the VT shall blank (i.e. fill with the background colour) the associated Key Cells so that the Key objects are not visible and cannot be activated by the operator. A change in state of the availability option can occur at runtime.

Working Set designers shall be aware that Keys might or might not be available and since the operator is in control of what is mapped on each User-Layout Data Mask, Window Mask objects shall not depend on the presence of a particular, or set of particular, Key Groups.

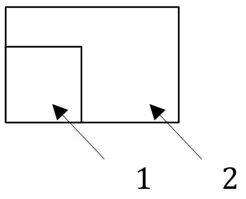


Key

1	Key Group containing 1 Key
2	Key Group containing 2 Keys
3	Key Group containing 3 Keys

Figure 29 — User-Layout Data Mask with 6 Key Cells — Example

Working Set designers should ensure that Keys can be recognized to be part of a specific Working Set (e.g. displaying just a text string “STOP” on a Key might lead to confusion by the operator if a particular configuration uses 3 “STOP” Keys in the same User-Layout Soft Key Mask). To create a consistent look and feel, the key layout shown in Figure 30 is recommended.



Key

1	Implement identifier (sized to 60 % of the height of a standard Soft Key designator (rounded down) and width equal to height) and anchored in the bottom left corner of the designator area
2	Key object in a Key Group is the size of a Soft Key designator (uses all remaining pixels in the standard designator area)

Figure 30 — Key object in a Key Group indicating Working Set — Example

4.7.10 Key Cell Size and Borders

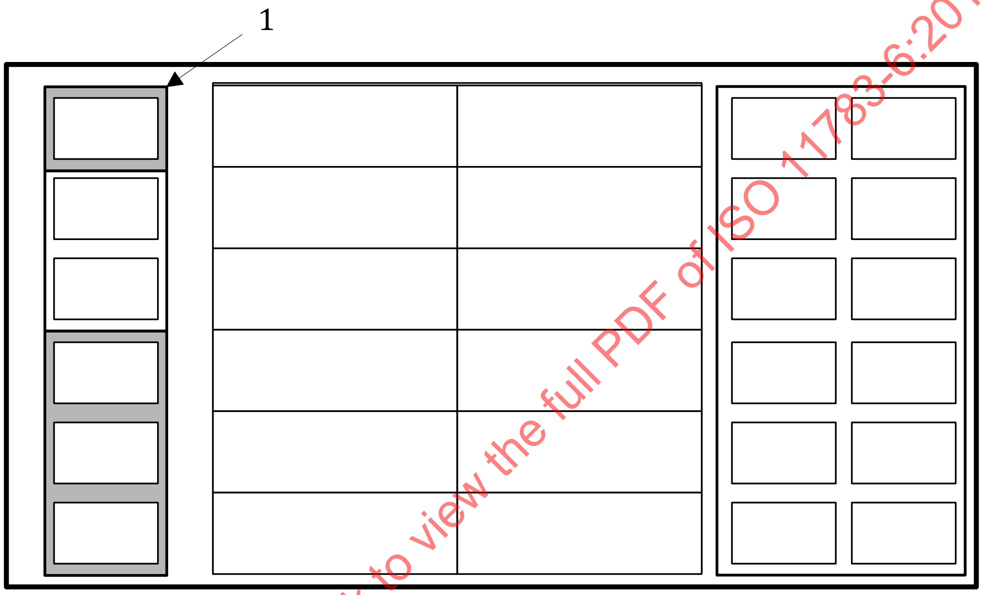
The same drawing rules that apply to Soft Key Mask objects also apply to User-Layout Soft Key Mask objects. A Key Cell size is the same size as a normal Soft Key designator.

4.7.11 Key Group Scaling

Similar to other objects in the Object Pool, scaling of the Key Group object's children is the responsibility of the Working Set.

4.7.12 Using Key Group Objects outside of User-Layout Soft Key Masks

The VT can optionally use Key Group objects outside of User-Layout Soft Key Mask, if supported by the manufacturer's VT display design. Working Sets shall be aware that Key Group objects can be used outside of User-Layout Soft Key Mask and therefore shall monitor the VT On User-Layout Hide/Show message and refresh keys that are visible regardless of the source address specified in the VT Status message.



Key

1 3 Key Group Objects outside of User-Layout Soft Key Masks

Figure 31 — Key Group Objects outside of User-Layout Data Mask — Example

4.7.13 Operator Inputs

It is possible that input objects from several Working Sets can be on screen at the same time. The usual navigation and data input rules apply with the following exceptions:

a) Select Input Object commands from any Working Set are not acted upon when a User-Layout Data Mask is displayed because the VT is the owner of the Data Mask. In this case, the VT shall send a Select Input Object response with an error indicated.

b) Navigation order is VT-proprietary.

c) Macros associated with input object events shall be executed but Working Set designers need to be aware that there might be no visible effect since the Working Set is not the active Working Set. For example, objects acted upon or made visible by the macro might not be on screen and Change Active Mask commands selecting a Data Mask will have no effect because the Working Set is not the active Working Set.

d) When an input object in a Window Mask is activated by the operator, the VT shall set Byte 2 of the VT Status message to FF_{16}, FE_{16}, or the VTs source address. If Working Set Data Mask(s) are also visible, the indicator of the active Data Mask shall be removed since the VT is now the active Working Set. Whether or not an "active working set" indicator is displayed around the Window

Mask is proprietary to the VT design. The VT shall also use the VT On user-Layout Hide/Show message to inform the Working Set about this change (see 4.6.8).

4.7.14 Refreshing On Screen Data

Whenever a Window Mask object or Key Group object is on screen, it shall be the responsibility of the Working Set to refresh values and objects as required. If a Working Set times out, the connection management rules apply and the VT shall remove from the screen, any Window Mask objects and Key Group objects that belong to the affected Working Set.

In order to refresh on screen objects, the Working Sets need to be made aware of what Window Mask objects and Key Group objects are visible. This is accomplished by the VT On User-Layout Hide/Show message (see H.20). This message shall be sent by the VT for each Window Mask and Key Group object that has been displayed or removed from the display.

NOTE The VT Status message is not used to determine when to update Window Mask or Key Group objects.

4.7.15 Look and Feel

4.7.15.1 User-Layout Data Mask Look and Feel

When mixing Window Mask objects from several Working Sets, general look and feel can quickly become a problem for the operator since objects might not line up vertically and horizontally and might use different colour schemes and fonts. Therefore restrictions are required for the design of the windows represented by the Window Mask objects and their children. In addition, the VT controls the look and feel on its User-Layout Data Masks. The strategy is that the Working Set supplies the superset of attributes that the VT might need to render the Window Mask objects but the VT decides what is displayed and where it is displayed in the Window Mask object assuming the window type is greater than zero. The following rules and guidelines shall apply:

a) When the Window Type is not the Free Form Window, the VT determines the background colour and transparency of the window. If required the Working Set designer can query the VT’s background colour with the Get Window Mask Data message.

b) Window contents should be designed as described in the following clauses.

c) The VT can ignore any visual formatting attributes in the Working Set’s supplied object references where the Window Type is not the Free Form Window.

4.7.15.2 Window Title and Window Title Font Attributes

The VT can optionally use this string for a title inside the Window Mask. Formatting is proprietary to the VT designer. The VT shall always display either or both of the Window Title and Window Icon as these elements are considered to be functionally equivalent and describe the contents of this window.

4.7.15.3 Window Icon size and shape

The VT can optionally use the Window Icon attribute inside the Window Mask. Positioning is proprietary to the VT designer, subject to the rules below. The Window Icon Area shall be square and shall be 90 % of the height of a single Window Cell, rounded down.

EXAMPLE In a 200x200 Data Mask Area, the Window Cell size is 100 pixels wide by 33 pixels high. The standard Window Icon Area is then:

Icon Height = 33 × 0.90 = 29 pixels

Icon Width = Icon Height = 29 pixels

This provides for room for a window border and some white space outside the Window Icon Area. Using the above information, Working Set designers can pre-scale the Window Mask Icon Picture Graphic object at design time or set the scale, via the width attribute, at run-time. It is permissible for the Working Set to supply an icon that is smaller or larger than the Window Icon Area. It is also permissible to provide an icon in an aspect ratio different from the square aspect ratio of the Window Icon Area. The VT can position the icon as desired if the icon dimension (X or Y) is smaller than the Window Icon Area. The VT shall centre and clip the icon if the icon dimension (X or Y) is larger than the Window Icon Area. These rules apply independently to the X and Y dimensions.

Working Set designers should supply an icon that clearly represents not only the specific function being displayed but also a representation of the Working Set so that the source of the data are evident to the operator.

4.7.15.4 Formatting

For window types greater than zero, the VT look and feel design shall ensure that at least the minimum number of characters are displayed for numeric and string value fields and shall obey the numeric scaling and numeric formatting attributes supplied by the Working Set. Specifically, this includes the options (bits 1, 2 and 3), variable reference, value, offset, scale, number of decimals and format attributes. Look and feel attributes such as justification, colour and other formatting attributes can be ignored by the VT to achieve its desired look and feel. More information on field lengths is given in the section on Window Mask Window Type (see B.19.2).

4.7.16 Uploading New Window Mask and Key Group objects

Uploading completely new objects (as long as the object type does not change) at run-time is permitted by Annex C of this document. In terms of Window Mask and Key Group objects, there are several cases that shall be considered and managed properly by the VT design as identified in Table 7.

Table 7 — VT Behaviour When New Window Mask or Key Group Object is Uploaded

Event	VT Behaviour
A new Window Mask or Window Mask child is uploaded that creates a simple change in appearance (background colour, option change, child change etc.).	If the object is visible, the VT shall refresh it.
A new Window Mask is uploaded that changes its availability from available to not available	If the object is visible, the VT shall blank (i.e. fill with the background colour) the area occupied by the object so that is it removed from the screen.
A new Window Mask is uploaded that changes its size	If the size decreases and the Window Mask is visible, the VT shall refresh the object and all window cells that it originally occupied. It shall also blank (i.e. fill with the background colour), all window cells that are no longer used by this object. The VT shall adjust the stored mapping to reflect the new unused window cells. If the size increases, the VT shall determine if the object still fits on screen in its current position in the grid and whether or not empty window cells are available in the extended positions to accommodate the new object. If yes, the VT shall refresh the object and adjust the stored mapping (for occupied window cells). If no, the VT shall automatically remove the window from the stored mapping and, if visible, shall blank (i.e. fill with the background colour) the window cells that it originally occupied.
A new Window Mask is uploaded that changes its window type	If the object is visible, the VT shall refresh it.

Table 7 *(continued)*

Event	VT Behaviour
A new Key Group object is uploaded that creates a simple change in appearance (option change, child change etc.)	If the object is visible, the VT shall refresh it.
A new Key Group object is uploaded that changes its availability from available to not available	If the object is visible, the VT shall blank (i.e. fill with the background colour) the area occupied by the object so that is it removed from the screen. The positions of other mapped Key Group objects shall not be effected.
A new Key Group object is uploaded that changes the number of keys in the group	If the number of keys in the group decreases, and the object is visible, the VT shall refresh the object and blank (i.e. fill with the background colour) and key positions that are no longer used. The VT shall adjust the stored mapping to reflect the new unused key positions. The positions of other mapped Key Group objects shall not be effected. If the number of keys in the group increases, the VT shall determine if there are enough empty key positions on the same page to accommodate the new object, without changing the position of the Key Group object. If yes, the VT shall extend the mapping of the object (for occupied key positions) and refresh the Key Group object if visible. If no, the VT shall automatically remove the Key Group from the mapping and, if the object is visible, shall blank (i.e. fill with the background colour) the key positions that it originally occupied. The positions of other mapped Key Group objects shall not be effected.

4.8 Colour Controls

The Working Set has various means to control the on-screen presentation by manipulating the colour palette being used for that individual Working Set.

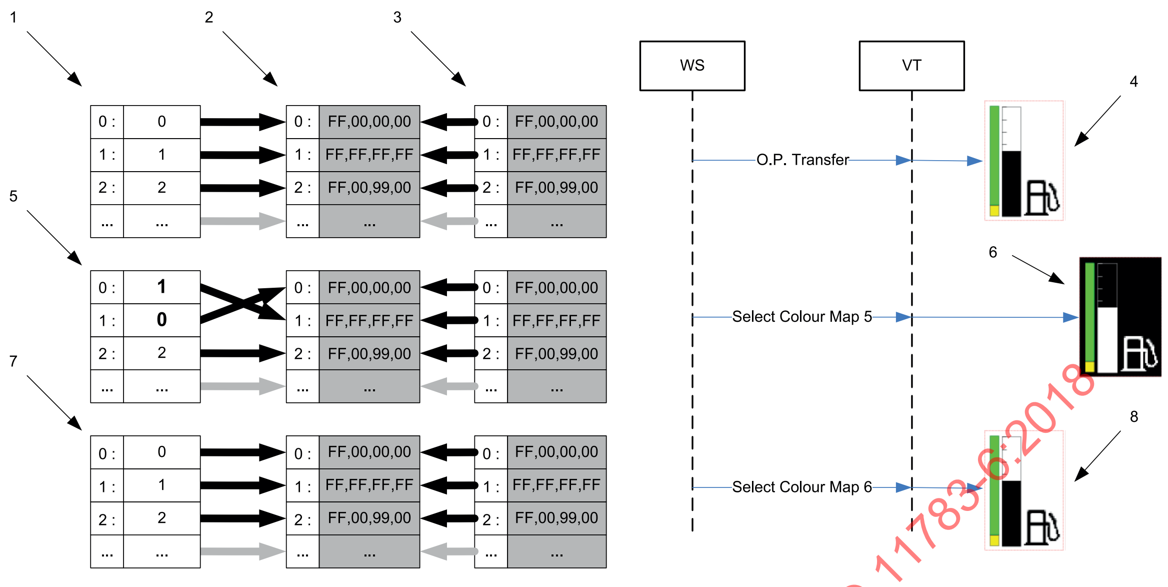
By default, the VT uses the VT standard colour palette (see A.3) which defines 256 colours, of which 24 are proprietary.

In VT version 4 and later, the Colour Map object (see B.17) can be used to alter the relationship between the colour attribute in a VT object and the index into the VT standard colour palette (see Figure 32).

In VT version 6 and later, the Colour Palette object (see B.26) can be used to redefine the colours in the colour palette (see Figure 33). The Colour Map object and the Colour Palette object apply individually to the object pool that contains these objects, and do not alter the presentation of other object pools.

The initial colour mapping as defined by the VT standard colour palette can be overridden with the Working Set Special Controls object (see B.29).

After initialization, runtime changes to the presentation colours can be performed using the Select Colour Map or Palette command (see F.60).



Key

1 VT default colour map

2 WS ARGB Colour Palette (initialized from Standard VT ARGB Colour Palette)

3 Standard VT ARGB Colour Palette

4 example presentation

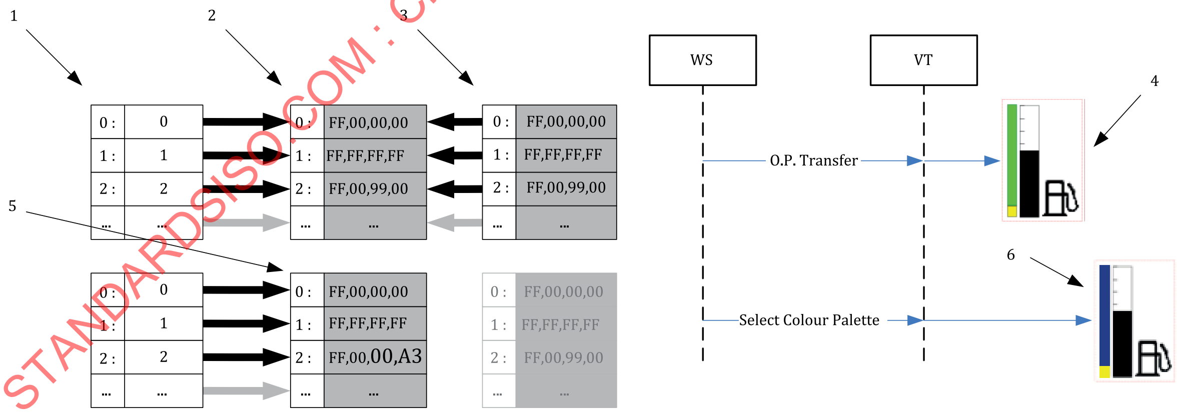
5 WS provided Colour Map object (reverses colour 0 and 1)

6 example presentation after Select Colour Map (5) command

7 WS provided Colour Map object (matches VT default colour map)

8 example presentation after Select Colour Map (6) command (restores to default)

Figure 32 — Colour Map object reverses and then restores colours — Example



Key

1 VT default colour map

2 WS ARGB Colour Palette (initialized from Standard VT ARGB Colour Palette)

3 Standard VT ARGB Colour Palette

4 example presentation

5 WS defined Colour Palette object (selected colour palette redefines only colour 2)

6 example presentation after selecting Colour Palette object

Figure 33 — Colour Palette object redefines colours — Example

Annex A
(normative)

Object, event, colour and command codes

A.1 Object types

A.1.1 General

This part of ISO 11783 takes an object-oriented approach. The VT shall be capable of managing the set of objects given in Table A.1, which includes the requirement to parse the objects, even if the object is not functionally supported. Each Working Set using the services of the VT defines an object pool that is a collection of the objects given. Each object has a specific, well-defined behaviour and a specific set of attributes.

Table A.1 — Virtual terminal objects

Object	Type ID	Description
Top level objects		
Working Set object	0_{10}	Top level object that describes an implement's ECU or group of ECUs (Working Set). Each Working Set is required to define one, and only one, Working Set object.
Data Mask object	1_{10}	Top level object that contains other objects. A Data Mask is activated by a Working Set to become the active set of objects on the VT display.
Alarm Mask object	2_{10}	Top level object that contains other objects. Describes an alarm display.
Container object	3_{10}	Used to group objects.
Window Mask object[e]	34_{10}	Top level object that contains other objects. The Window Mask is activated by the VT.
Key objects		
Soft Key Mask object	4_{10}	Top level object that contains Key objects.
Key object	5_{10}	Used to describe a Soft Key.
Button object	6_{10}	Used to describe a Button control.
Key Group object[e]	35_{10}	Top level object that contains Key objects.
Input field objects		
Input Boolean object	7_{10}	Used to input a TRUE/FALSE type input.
Input String object	8_{10}	Used to input a character string.
Input Number object	9_{10}	Used to input an integer or float numeric.
Input List object	10_{10}	Used to select an item from a pre-defined list.
Output field objects		
Output String object	11_{10}	Used to output a character string.
Output Number object	12_{10}	Used to output an integer or float numeric.
Output List object[a]	37_{10}	Used to output a list item.
Output shape objects		
Output Line object	13_{10}	Used to output a line.
Output Rectangle object	14_{10}	Used to output a rectangle or square.

Table A.1 *(continued)*

Object	Type ID	Description
Output Ellipse object	15_{10}	Used to output an ellipse or circle.
Output Polygon object	16_{10}	Used to output a polygon.
Output graphic objects		
Output Meter object	17_{10}	Used to output a meter.
Output Linear Bar Graph object	18_{10}	Used to output a linear bar graph.
Output Arched Bar Graph object	19_{10}	Used to output an arched bar graph.
Graphics Context object[e]	36_{10}	Used to output a graphics context.
Animation object[f]	44_{10}	The Animation object is used to display simple animations.
Picture graphic object		
Picture Graphic object	20_{10}	Used to output a picture graphic (bitmap).
Graphic Data object[g]	46_{10}	Used to define the data for a graphic image.
Scaled Graphic object[g]	48_{10}	Used to display a scaled representation of a graphic object.
Variable objects		
Number Variable object	21_{10}	Used to store a 32-bit unsigned integer value.
String Variable object	22_{10}	Used to store a fixed length string value.
Attribute Objects		
Font Attributes object	23_{10}	Used to group font based attributes. Can only be referenced by other objects.
Line Attributes object	24_{10}	Used to group line based attributes. Can only be referenced by other objects.
Fill Attributes object	25_{10}	Used to group fill based attributes. Can only be referenced by other objects.
Input Attributes object	26_{10}	Used to specify a list of valid characters. Can only be referenced by input field objects.
Extended Input Attributes object[a]	38_{10}	Used to specify a list of valid WideChars. Can only be referenced by Input Field Objects.
Colour Map object[e]	39_{10}	Used to specify a colour table object.
Object Label Reference List object[a]	40_{10}	Used to specify an object label.
Colour Palette object[g]	45_{10}	Used to specify a colour palette
Working Set Special Controls object[g]	47_{10}	Used to provide special controls over colour maps and palettes
Pointer object		
Object Pointer object	27_{10}	Used to reference another object.
External Object Definition object[f]	41_{10}	Used to list the objects that can be referenced from another Working Set
External Reference NAME object[f]	42_{10}	Used to identify the WS Master of a Working Set that can be referenced
External Object Pointer object[f]	43_{10}	Used to reference an object in another Working Set
Macro object		
Macro object	28_{10}	Special object that contains a list of commands that can be executed in response to an event. Macros can be referenced by other objects. Version 4 and later Working Sets can use the Execute Macro command. Version 5 and later works Sets can use the Execute Extended Macro command.

Table A.1 *(continued)*

Object	Type ID	Description
Auxiliary control		
Auxiliary Function Type 1 object (ignored)[c]	29_{10}	The Auxiliary Function Type 1 object defines the designator and function type for an Auxiliary Function. The object is used strictly in the Auxiliary Control screen which is proprietary to the VT.
Auxiliary Input Type 1 object (ignored)[c]	30_{10}	The Auxiliary Input Type 1 object defines the designator, key number, and function type for an auxiliary input. The object is used strictly in the Auxiliary Control screen which is proprietary to the VT.
Auxiliary Function Type 2 object[b]	31_{10}	The Auxiliary Function Type 2 object defines the designator and function type for an Auxiliary Function.
Auxiliary Input Type 2 object[b]	32_{10}	The Auxiliary Input Type 2 object defines the designator, key number, and function type for an Auxiliary Input.
Auxiliary Control Designator Type 2 Object Pointer[b]	33_{10}	Used to reference Auxiliary Input Type 2 object or Auxiliary Function Type 2 object.
Proprietary Objects		
Manufacturer Defined Objects[d]	240_{10}—254_{10}	Manufacturer defined objects should not be sent to any other Vendors VT (see 4.6.24).
Reserved Objects		
Reserved	49_{10}—239_{10}	Reserved for future use.
Reserved	255_{10}	Reserved for future use (see D.14).

[a] Version 4 and later VTs support these objects.

[b] Version 3 and later VTs support these objects.

[c] Version 3 and later VTs parse these objects for compatibility, but they are not functionally supported.

[d] Version 4 and later VTs support proprietary objects that should not be used between ECUs and VTs with different manufacturer codes.

[e] Version 4 and later VTs parse these objects for compatibility because they are optional and might not be functionally supported (see D.14).

[f] Version 5 and later VTs support these objects.

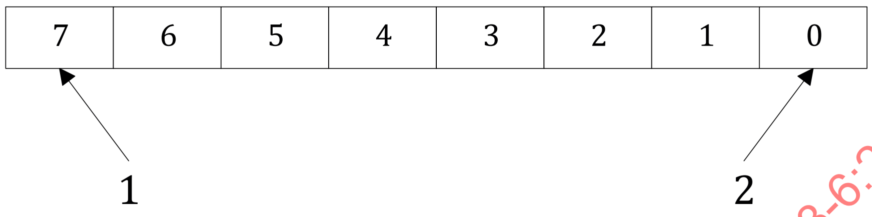
[g] Version 6 and later VTs support this object.

A.1.2 Nomenclature

The following data types and nomenclature are used in the object definitions in Annex B.

[ ]	When surrounding an AID number this indicates that it is a read-only attribute and is accessible with the Get Attribute Value message. AIDs that are explicitly defined without square brackets are writable with the Change Attribute command.
Array	A sequence of 1 byte unsigned integer values of a defined length.
Bitmask	A set of logical bit values. Size is 1 byte. Bitmasks always have Bit 0 defined as the least significant bit (see Figure A.1).
Boolean	Logical TRUE (1) or FALSE (0). Size is 1 byte.
Byte	Signed or unsigned integer numeric value with a size of exactly 1 byte.
Float	IEEE 754-1985 standard 32-bit floating point numeric value. Size is 4 bytes.

Integer	Signed or unsigned integer numeric value. Possible sizes are 1, 2 or 4 bytes.
String	Zero or more characters composed of either the primitive type “Char” or the primitive type WideChar. String length is variable.
Length	The size of an object, always expressed as a count of the Bytes required to hold the object.



Key

1 most significant

2 least significant

Figure A.1 — Bit positions in a bitmask

A.1.3 Object relationships

The objects of an object pool are arranged in a hierarchy, where parent objects contain child objects (see 4.6.1.3).

Some of the child objects can also contain objects, and then they can become parent objects to their own child-objects. Table A.2 shows the containment rules of objects within the object pool hierarchy.

Table A.2 — Allowed hierarchical relationships of objects

Child objects	Parent objects																	
	Working Set object	Data Mask object	Alarm Mask object	Container object	Window Mask object	Soft Key Mask object	Key object	Button object	Key Group object	Input List object	Output List object	Auxiliary Function Type 1 object	Auxiliary Input Type 1 object	Auxiliary Function Type 2 object	Auxiliary Input Type 2 object	Object Label graphic representation	Object Label Reference List object	Animation object
Working Set object		3	3	3	4		4	4		4	4						4	
Data Mask object																	4	
Alarm Mask object																	4	
Container object	4	2	2	2	4		2	2		4	4			3	3	4	4	5
Window Mask object																	4	
Soft Key Mask object																	4	
Key object						2			4								4	
Button object		2		2	4						4						4	
Key Group object																	4	
Input Boolean object		2		2	4						4						4	
Input String object		2		2	4						4						4	
Input Number object		2		2	4						4						4	
Input List object		2		2	4						4						4	
Output String object	2	2	2	2	4		2	2		2	4	2	2	3	3	4	4	5
Output Number object	2	2	2	2	4		2	2		2	4	2	2	3	3	4	4	5
Output List object	4	4	4	4	4		4	4		4	4			4	4	4	4	5
Output Line object	2	2	2	2	4		2	2		4	4	2	2	3	3	4	4	5
Output Rectangle object	2	2	2	2	4		2	2		4	4	2	2	3	3	4	4	5
Output Ellipse object	2	2	2	2	4		2	2		4	4	2	2	3	3	4	4	5
Output Polygon object	2	2	2	2	4		2	2		4	4	2	2	3	3	4	4	5
Output Meter object	4	2	2	2	4		4	4		4	4			3	3	4	4	5
Output Linear Bar Graph object	4	2	2	2	4		4	4		4	4			3	3	4	4	5

Table A.2 *(continued)*

Child objects	Parent objects																	
	Working Set object	Data Mask object	Alarm Mask object	Container object	Window Mask object	Soft Key Mask object	Key object	Button object	Key Group object	Input List object	Output List object	Auxiliary Function Type 1 object	Auxiliary Input Type 1 object	Auxiliary Function Type 2 object	Auxiliary Input Type 2 object	Object Label graphic representation	Object Label Reference List object	Animation object
Output Arched Bar Graph object	4	2	2	2	4		4	4		4	4			3	3	4	4	5
Graphics Context object	4	4	4	4	4		4	4		4	4			4	4	4	4	5
Animation object		5	5	5	5		5	5			5						5	
Picture Graphic object	2	2	2	2	4		2	2		2	4	2	2	3	3	4	4	5
Scaled Graphic object	6	6	6	6	6		6	6		6	6			6	6	6	6	6
Number Variable object																	5	
String Variable object																	5	
Font Attributes object																	5	
Line Attributes object																	5	
Fill Attributes object																	5	
Input Attributes object																	5	
Extended Input Attributes object																	5	
Colour Map object																	5	
Object Label Reference List object																	5	
Object Pointer object	4	2	2	2	4	2	2	2	4	4	4			3	3	4	5	5
External Object Definition object																	5	
External Reference NAME object																	5	
External Object Pointer object		5	5	5	5	5	5	5	5	5	5						5	

Table A.2 *(continued)*

Child objects	Parent objects																
	Working Set object	Data Mask object	Alarm Mask object	Container object	Window Mask object	Soft Key Mask object	Key object	Button object	Key Group object	Input List object	Output List object	Auxiliary Function Type 1 object	Auxiliary Input Type 1 object	Auxiliary Function Type 2 object	Auxiliary Input Type 2 object	Object Label graphic representation	Object Label Reference List object	Animation object
Macro object																	5	
Auxiliary Function Type 1 object																	5	
Auxiliary Input Type 1 object																	5	
Auxiliary Function Type 2 object		3	3	3	6						6						4	
Auxiliary Input Type 2 object		3	3	3	6						6						4	
Auxiliary Control Designator Type 2 Object Pointer		3	3	3	4						4						5	
Colour Palette object																	6	
Graphic Data object																	6	
Working Set Special Controls object																	6	

NOTE 1 The numbers in the table above denote Child Objects that can be contained within a Parent Object in the indicated VT version including later versions.

NOTE 2 Containment rules of the parent override containment rules of the child

NOTE 3 Object pointed to by Object Pointer cannot violate the containment rules of the parent hierarchy

A.2 Event types

Table A.3 presents the events defined by this part of ISO 11783. An event ID is assigned to each so that events can be uniquely associated with a Macro object to execute when the event occurs. VT behaviour specific to each object is defined in event tables given with each object.

Table A.3 — Event summary

Event name	Event ID 8-bit Macro 16-bit Macro[c]	Event occurs when:
Reserved	0_{10}	Reserved
On activate	1_{10}	Working Set is made active.
On deactivate	2_{10}	Working Set is made inactive.
On show	3_{10}	For Container objects, triggered by the hide/show command, with “show” indicated; for mask objects, when the mask is made visible on the display.
On hide	4_{10}	For Container objects, triggered by the hide/show command, with “hide” indicated; for mask objects, when the mask is removed from the display.
On refresh	N/A	An object that is already on display is redrawn (Macros cannot be associated with this event so no event ID is defined).
On enable	5_{10}	Input object is enabled (only enabled input objects can be navigated to). An Animation object is enabled for animation.
On disable	6_{10}	Input object is disabled (only enabled input objects can be navigated to). An Animation object is disabled for animation.
On Change Active Mask	7_{10}	Change Active Mask command.
On Change Soft Key Mask	8_{10}	Change Soft Key Mask command.
On Change Attribute	9_{10}	Change Attribute command.
On Change Background Colour	10_{10}	Change Background Colour command.
On Change Font Attributes	11_{10}	Change Font Attributes command.
On Change Line Attributes	12_{10}	Change Line Attributes command.
On Change Fill Attributes	13_{10}	Change Fill Attributes command.
On Change Child Location	14_{10}	Change Child Location command.
On Change Size	15_{10}	Change Size command.
On Change Value	16_{10}	Change Numeric Value command or Change String Value command.
On Change Priority	17_{10}	Change Priority command.
On Change End Point	18_{10}	Change End Point command.

[a] If two or more input objects reference the same variable object and one of the input objects is modified (thereby changing the value of the variable object), the OnEntryOfAValue and conditionally the OnEntryOfANewValue Macros are executed for the modified input object only.

[b] VT version 4 and prior.

[c] VT version 5 and later.

[d] If an object already has focus and the operator opens or closes it, then the OnInputFieldSelection macro shall not be executed. For the Select Input Object command the rules of 4.6.22.1 apply, i.e. any valid command shall cause an event.

Table A.3 *(continued)*

Event name	Event ID 8-bit Macro 16-bit Macro[c]	Event occurs when:
On Input Field Selection	19_{10}[d]	The input field, Key or Button has received focus, operator has navigated onto the input field or Button or the VT has received the Select Input Object command.
On Input Field Deselection	20_{10}[d]	The input field, Key or Button has lost focus, operator has navigated off of the input field or Button or the VT has received the Select Input Object command.
On ESC	21_{10}	Input aborted on an input field either by the operator or the Working Set.
On entry of a value	22_{10} [a]	Operator completes entry by activating the ENTER means — value does not have to change.
On entry of a new value	23_{10}[a]	Operator completes entry by activating the ENTER means — value has changed.
On key press	24_{10}	A Soft Key or Button is pressed.
On key release	25_{10}	A Soft Key or Button is released.
On Change Child Position	26_{10}	Change Child Position command.
On pointing event press	27_{10}[c]	Operator touches/clicks an area that causes a pointing event
On pointing event release	28_{10}[c]	Operator touch/click is released
Reserved	27_{10}–239_{10}[b] 29_{10}–239_{10}[c]	Reserved
Proprietary Events	240_{10}–254_{10}[b]	Proprietary events should not be used between ECUs and VTs with different manufacturer codes.
Reserved	255_{10}[b]	Reserved
Use Extended Macro Reference	255_{10}[c]	This is not an event. When value is found in the event list of an object, it indicates that a 16-bit Macro Object ID reference is used (see 4.6.22.3).

[a] If two or more input objects reference the same variable object and one of the input objects is modified (thereby changing the value of the variable object), the OnEntryOfAValue and conditionally the OnEntryOfANewValue Macros are executed for the modified input object only.

[b] VT version 4 and prior.

[c] VT version 5 and later.

[d] If an object already has focus and the operator opens or closes it, then the OnInputFieldSelection macro shall not be executed. For the Select Input Object command the rules of 4.6.22.1 apply, i.e. any valid command shall cause an event.

A.3 VT standard colour palette

The VT Standard colour palette is shown in Table A.4.

In VT version 6 and later, the colour palette includes an alpha channel value for each colour. The alpha channel value is configured by default to be opaque (alpha 255), which is then fully backward compatible with VT version 5 and prior, however the colour palette can be redefined by each individual Working Set using the Colour Palette object (see 4.6.13 and B.26). Additionally, the colour map and colour palette can be defined in the object pool in a manner that activates the alternate colour scheme when the pool is loaded (see B.29).

In VT version 4 or later, the active Colour Map can be altered (see B.17 and F.60).

Table A.4 — Standard VT ARGB colour palette

Index	A,R,G,B value	Index	A,R,G,B value	Index	A,R,G,B value	Index	A,R,G,B value
0 (Black)	FF,00,00,00[a, b]	64	FF,33,66,00	128	FF,99,00,CC	192	FF,CC,FF,66
1 (White)	FF,FF,FF,FF[a, b]	65	FF,33,66,33	129	FF,99,00,FF	193	FF,CC,FF,99
2 (Green)	FF,00,99,00[b]	66	FF,33,66,66	130	FF,99,33,00	194	FF,CC,FF,CC
3 (Teal)	FF,00,99,99[b]	67	FF,33,66,99	131	FF,99,33,33	195	FF,CC,FF,FF
4 (Maroon)	FF,99,00,00[b]	68	FF,33,66,CC	132	FF,99,33,66	196	FF,FF,00,00
5 (Purple)	FF,99,00,99[b]	69	FF,33,66,FF	133	FF,99,33,99	197	FF,FF,00,33
6 (Olive)	FF,99,99,00[b]	70	FF,33,99,00	134	FF,99,33,CC	198	FF,FF,00,66
7 (Silver)	FF,CC,CC,CC[b]	71	FF,33,99,33	135	FF,99,33,FF	199	FF,FF,00,99
8 (Grey)	FF,99,99,99[b]	72	FF,33,99,66	136	FF,99,66,00	200	FF,FF,00,CC
9 (Blue)	FF,00,00,FF[b]	73	FF,33,99,99	137	FF,99,66,33	201	FF,FF,00,FF
10 (Lime)	FF,00,FF,00[b]	74	FF,33,99,CC	138	FF,99,66,66	202	FF,FF,33,00
11 (Cyan)	FF,00,FF,FF[b]	75	FF,33,99,FF	139	FF,99,66,99	203	FF,FF,33,33
12 (Red)	FF,FF,00,00[b]	76	FF,33,CC,00	140	FF,99,66,CC	204	FF,FF,33,66
13 (Magenta)	FF,FF,00,FF[b]	77	FF,33,CC,33	141	FF,99,66,FF	205	FF,FF,33,99
14 (Yellow)	FF,FF,FF,00[b]	78	FF,33,CC,66	142	FF,99,99,00	206	FF,FF,33,CC
15 (Navy)	FF,00,00,99[b]	79	FF,33,CC,99	143	FF,99,99,33	207	FF,FF,33,FF
16	FF,00,00,00	80	FF,33,CC,CC	144	FF,99,99,66	208	FF,FF,66,00
17	FF,00,00,33	81	FF,33,CC,FF	145	FF,99,99,99	209	FF,FF,66,33
18	FF,00,00,66	82	FF,33,FF,00	146	FF,99,99,CC	210	FF,FF,66,66
19	FF,00,00,99	83	FF,33,FF,33	147	FF,99,99,FF	211	FF,FF,66,99
20	FF,00,00,CC	84	FF,33,FF,66	148	FF,99,CC,00	212	FF,FF,66,CC
21	FF,00,00,FF	85	FF,33,FF,99	149	FF,99,CC,33	213	FF,FF,66,FF
22	FF,00,33,00	86	FF,33,FF,CC	150	FF,99,CC,66	214	FF,FF,99,00
23	FF,00,33,33	87	FF,33,FF,FF	151	FF,99,CC,99	215	FF,FF,99,33
24	FF,00,33,66	88	FF,66,00,00	152	FF,99,CC,CC	216	FF,FF,99,66
25	FF,00,33,99	89	FF,66,00,33	153	FF,99,CC,FF	217	FF,FF,99,99
26	FF,00,33,CC	90	FF,66,00,66	154	FF,99,FF,00	218	FF,FF,99,CC
27	FF,00,33,FF	91	FF,66,00,99	155	FF,99,FF,33	219	FF,FF,99,FF
28	FF,00,66,00	92	FF,66,00,CC	156	FF,99,FF,66	220	FF,FF,CC,00
29	FF,00,66,33	93	FF,66,00,FF	157	FF,99,FF,99	221	FF,FF,CC,33
30	FF,00,66,66	94	FF,66,33,00	158	FF,99,FF,CC	222	FF,FF,CC,66
31	FF,00,66,99	95	FF,66,33,33	159	FF,99,FF,FF	223	FF,FF,CC,99
32	FF,00,66,CC	96	FF,66,33,66	160	FF,CC,00,00	224	FF,FF,CC,CC
33	FF,00,66,FF	97	FF,66,33,99	161	FF,CC,00,33	225	FF,FF,CC,FF
34	FF,00,99,00	98	FF,66,33,CC	162	FF,CC,00,66	226	FF,FF,FF,00
35	FF,00,99,33	99	FF,66,33,FF	163	FF,CC,00,99	227	FF,FF,FF,33
36	FF,00,99,66	100	FF,66,66,00	164	FF,CC,00,CC	228	FF,FF,FF,66
37	FF,00,99,99	101	FF,66,66,33	165	FF,CC,00,FF	229	FF,FF,FF,99
38	FF,00,99,CC	102	FF,66,66,66	166	FF,CC,33,00	230	FF,FF,FF,CC
39	FF,00,99,FF	103	FF,66,66,99	167	FF,CC,33,33	231	FF,FF,FF,FF
40	FF,00,CC,00	104	FF,66,66,CC	168	FF,CC,33,66	232	Proprietary

[a] Monochrome (0 and 1).

[b] 16-colour mode (0 to 15).

Table A.4 *(continued)*

Index	A,R,G,B value	Index	A,R,G,B value	Index	A,R,G,B value	Index	A,R,G,B value
41	FF,00,CC,33	105	FF,66,66,FF	169	FF,CC,33,99	233	Proprietary
42	FF,00,CC,66	106	FF,66,99,00	170	FF,CC,33,CC	234	Proprietary
43	FF,00,CC,99	107	FF,66,99,33	171	FF,CC,33,FF	235	Proprietary
44	FF,00,CC,CC	108	FF,66,99,66	172	FF,CC,66,00	236	Proprietary
45	FF,00,CC,FF	109	FF,66,99,99	173	FF,CC,66,33	237	Proprietary
46	FF,00,FF,00	110	FF,66,99,CC	174	FF,CC,66,66	238	Proprietary
47	FF,00,FF,33	111	FF,66,99,FF	175	FF,CC,66,99	239	Proprietary
48	FF,00,FF,66	112	FF,66,CC,00	176	FF,CC,66,CC	240	Proprietary
49	FF,00,FF,99	113	FF,66,CC,33	177	FF,CC,66,FF	241	Proprietary
50	FF,00,FF,CC	114	FF,66,CC,66	178	FF,CC,99,00	242	Proprietary
51	FF,00,FF,FF	115	FF,66,CC,99	179	FF,CC,99,33	243	Proprietary
52	FF,33,00,00	116	FF,66,CC,CC	180	FF,CC,99,66	244	Proprietary
53	FF,33,00,33	117	FF,66,CC,FF	181	FF,CC,99,99	245	Proprietary
54	FF,33,00,66	118	FF,66,FF,00	182	FF,CC,99,CC	246	Proprietary
55	FF,33,00,99	119	FF,66,FF,33	183	FF,CC,99,FF	247	Proprietary
56	FF,33,00,CC	120	FF,66,FF,66	184	FF,CC,CC,00	248	Proprietary
57	FF,33,00,FF	121	FF,66,FF,99	185	FF,CC,CC,33	249	Proprietary
58	FF,33,33,00	122	FF,66,FF,CC	186	FF,CC,CC,66	250	Proprietary
59	FF,33,33,33	123	FF,66,FF,FF	187	FF,CC,CC,99	251	Proprietary
60	FF,33,33,66	124	FF,99,00,00	188	FF,CC,CC,CC	252	Proprietary
61	FF,33,33,99	125	FF,99,00,33	189	FF,CC,CC,FF	253	Proprietary
62	FF,33,33,CC	126	FF,99,00,66	190	FF,CC,FF,00	254	Proprietary
63	FF,33,33,FF	127	FF,99,00,99	191	FF,CC,FF,33	255	Proprietary

[a] Monochrome (0 and 1).

[b] 16-colour mode (0 to 15).

The VT standard colour palette is based on the standard 216 colour "web browser safe" palette used by Internet browsers. Hexadecimal RGB values of 00, 33, 66, 99, CC and FF are used giving a 6 × 6 × 6 = 216 colour cube. The colour palette is organized as follows. The first two colours at indices 0 and 1 are used in monochrome mode. The first 16 colours are used in 16 colour mode. In order to reduce search time during palette mapping on an object development tool, colours 16 to 231 are organized in sorted, ascending order. Colours 232 to 255 are proprietary to the VT design to extend the colour palette. For VT version 5 and prior, VT designers choosing a grey-scale implementation can map the 16 or 256 colour modes to shades of grey.

NOTE 256 colour mode does not actually give 256 unique colours because the first 16 colours are repeated elsewhere in the palette.

Colour and pixel values given in object attributes and bitmap data are an index into this palette table.

There are three defined colour modes. VT designs supporting higher modes shall also support lower modes, as follows:

a) monochrome only (black and some other colour, usually white) (valid colour codes 0 to 1);

b) 16-colour mode (VT shall support 16 colour and monochrome) (valid colour codes 0 to 15);

c) 256-colour mode (VT shall support all colours of the palette) (valid colour codes 0 to 255).

A.4 Command/parameter code summary

Table A.5 lists all defined messages with the corresponding function code.

Table A.5 — Command/parameter summary

Clause/ subclause	Message	PGN	Function (Decimal)	Function (Hex)	Allowed in macro	VT version[c]
C.2.3	Object pool transfer message	ECU to VT	17_{10}	11_{16}	No	2
C.2.4	End of Object Pool message	ECU to VT	18_{10}	12_{16}	No	2
C.2.5	End of Object Pool response	VT to ECU	18_{10}	12_{16}	No	2
D.2	Get Memory message	ECU to VT	192_{10}	$C0_{16}$	No	2
D.3	Get Memory response	VT to ECU	192_{10}	$C0_{16}$	No	2
D.4	Get Number of Soft Keys message	ECU to VT	194_{10}	$C2_{16}$	No	2
D.5	Get Number of Soft Keys response	VT to ECU	194_{10}	$C2_{16}$	No	2
D.6	Get Text Font Data message	ECU to VT	195_{10}	$C3_{16}$	No	2
D.7	Get Text Font Data response	VT to ECU	195_{10}	$C3_{16}$	No	2
D.8	Get Hardware message	ECU to VT	199_{10}	$C7_{16}$	No	2
D.9	Get Hardware response	VT to ECU	199_{10}	$C7_{16}$	No	2
D.10	Get Supported Widechars message	ECU to VT	193_{10}	$C1_{16}$	No	4
D.11	Get Supported WideChars response	VT to ECU	193_{10}	$C1_{16}$	No	4
D.12	Get Window Mask Data message	ECU to VT	196_{10}	$C4_{16}$	No	4
D.13	Get Window Mask Data response	VT to ECU	196_{10}	$C4_{16}$	No	4
D.14	Get Supported Objects message	ECU to VT	197_{10}	$C5_{16}$	No	4
D.15	Get Supported Objects response	VT to ECU	197_{10}	$C5_{16}$	No	4
D.16	Screen Capture command	ECU to VT	198_{10}	$C6_{16}$	No	6
D.17	Screen Capture response	VT to ECU	198_{10}	$C6_{16}$	No	6
D.18	Identify VT message	ECU to VT	187_{10}	BB_{16}	No	4
D.19	Identify VT response	VT to ECU	187_{10}	BB_{16}	No	4
E.2	Get Versions message	ECU to VT	223_{10}	DF_{16}	No	2
E.3	Get Versions response	VT to ECU	224_{10}	$E0_{16}$	No	2
E.4	Store Version command	ECU to VT	208_{10}	$D0_{16}$	No	2
E.5	Store Version response	VT to ECU	208_{10}	$D0_{16}$	No	2
E.6	Load Version command	ECU to VT	209_{10}	$D1_{16}$	No	2
E.7	Load Version response	VT to ECU	209_{10}	$D1_{16}$	No	2
E.8	Delete Version command	ECU to VT	210_{10}	$D2_{16}$	No	2
E.9	Delete Version response	VT to ECU	210_{10}	$D2_{16}$	No	2
E.10	Extended Get Versions message	ECU to VT	211_{10}	$D3_{16}$	No	5
E.11	Extended Get Versions response	VT to ECU	211_{10}	$D3_{16}$	No	5
E.12	Extended Store Version command	ECU to VT	212_{10}	$D4_{16}$	No	5
E.13	Extended Store Version response	VT to ECU	212_{10}	$D4_{16}$	No	5
E.14	Extended Load Version command	ECU to VT	213_{10}	$D5_{16}$	No	5
E.15	Extended Load Version response	VT to ECU	213_{10}	$D5_{16}$	No	5

[a] Reserved for future use.

[b] Proprietary commands should not be used between ECUs and VT with different manufacturer codes.

[c] Identifies the VT version at which the command was introduced, however some messages have been revised with later versions.

Table A.5 *(continued)*

Clause/ subclause	Message	PGN	Function (Decimal)	Function (Hex)	Allowed in macro	VT version[c]
F.16	Extended Delete Version command	ECU to VT	214_{10}	$D6_{16}$	No	5
F.17	Extended Delete Version response	VT to ECU	214_{10}	$D6_{16}$	No	5
F.2	Hide/Show Object command	ECU to VT	160_{10}	$A0_{16}$	Yes	2
F.3	Hide/Show Object response	VT to ECU	160_{10}	$A0_{16}$	No	2
F.4	Enable/Disable Object command	ECU to VT	161_{10}	$A1_{16}$	Yes	2
F.5	Enable/Disable Object response	VT to ECU	161_{10}	$A1_{16}$	No	2
F.6	Select Input Object command	ECU to VT	162_{10}	$A2_{16}$	Yes	2
F.7	Select Input Object response	VT to ECU	162_{10}	$A2_{16}$	No	2
F.8	ESC command	ECU to VT	146_{10}	92_{16}	No	2
F.9	ESC response	VT to ECU	146_{10}	92_{16}	No	2
F.10	Control Audio Signal command	ECU to VT	163_{10}	$A3_{16}$	Yes	2
F.11	Control Audio Signal response	VT to ECU	163_{10}	$A3_{16}$	No	2
F.12	Set Audio Volume command	ECU to VT	164_{10}	$A4_{16}$	Yes	2
F.13	Set Audio Volume response	VT to ECU	164_{10}	$A4_{16}$	No	2
F.14	Change Child Location command	ECU to VT	165_{10}	$A5_{16}$	Yes	2
F.15	Change Child Location response	VT to ECU	165_{10}	$A5_{16}$	No	2
F.16	Change Child Position command	ECU to VT	180_{10}	$B4_{16}$	Yes	2
F.17	Change Child Position response	VT to ECU	180_{10}	$B4_{16}$	No	2
F.18	Change Size command	ECU to VT	166_{10}	$A6_{16}$	Yes	2
F.19	Change Size response	VT to ECU	166_{10}	$A6_{16}$	No	2
F.20	Change Background Colour command	ECU to VT	167_{10}	$A7_{16}$	Yes	2
F.21	Change Background Colour response	VT to ECU	167_{10}	$A7_{16}$	No	2
F.22	Change Numeric Value command	ECU to VT	168_{10}	$A8_{16}$	Yes	2
F.23	Change Numeric Value response	VT to ECU	168_{10}	$A8_{16}$	No	2
F.24	Change String Value command	ECU to VT	179_{10}	$B3_{16}$	Yes	2
F.25	Change String Value response	VT to ECU	179_{10}	$B3_{16}$	No	2
F.26	Change End Point command	ECU to VT	169_{10}	$A9_{16}$	Yes	2
F.27	Change End Point response	VT to ECU	169_{10}	$A9_{16}$	No	2
F.28	Change Font Attributes command	ECU to VT	170_{10}	AA_{16}	Yes	2
F.29	Change Font Attributes response	VT to ECU	170_{10}	AA_{16}	No	2
F.30	Change Line Attributes command	ECU to VT	171_{10}	AB_{16}	Yes	2
F.31	Change Line Attributes response	VT to ECU	171_{10}	AB_{16}	No	2
F.32	Change Fill Attributes command	ECU to VT	172_{10}	AC_{16}	Yes	2
F.33	Change Fill Attributes response	VT to ECU	172_{10}	AC_{16}	No	2
F.34	Change Active Mask command	ECU to VT	173_{10}	AD_{16}	Yes	2
F.35	Change Active Mask response	VT to ECU	173_{10}	AD_{16}	No	2
F.36	Change Soft Key Mask command	ECU to VT	174_{10}	AE_{16}	Yes	2
F.37	Change Soft Key Mask response	VT to ECU	174_{10}	AE_{16}	No	2

[a] Reserved for future use.

[b] Proprietary commands should not be used between ECUs and VT with different manufacturer codes.

[c] Identifies the VT version at which the command was introduced, however some messages have been revised with later versions.

Table A.5 *(continued)*

Clause/ subclause	Message	PGN	Function (Decimal)	Function (Hex)	Allowed in macro	VT version[c]
F.38	Change Attribute command	ECU to VT	175_{10}	AF_{16}	Yes	2
F.39	Change Attribute response	VT to ECU	175_{10}	AF_{16}	No	2
F.40	Change Priority command	ECU to VT	176_{10}	$B0_{16}$	Yes	2
F.41	Change Priority response	VT to ECU	176_{10}	$B0_{16}$	No	2
F.42	Change List Item command	ECU to VT	177_{10}	$B1_{16}$	Yes	2
F.43	Change List Item response	VT to ECU	177_{10}	$B1_{16}$	No	2
F.44	Delete Object Pool command	ECU to VT	178_{10}	$B2_{16}$	No	2
F.45	Delete Object Pool response	VT to ECU	178_{10}	$B2_{16}$	No	2
F.46	Lock/Unlock Mask command	ECU to VT	189_{10}	BD_{16}	Yes	4
F.47	Lock/Unlock Mask response	VT to ECU	189_{10}	BD_{16}	No	4
F.48	Execute Macro command	ECU to VT	190_{10}	BE_{16}	Yes	4
F.49	Execute Macro response	VT to ECU	190_{10}	BE_{16}	No	4
F.50	Change Object Label command	ECU to VT	181_{10}	$B5_{16}$	Yes	4
F.51	Change Object Label response	VT to ECU	181_{10}	$B5_{16}$	No	4
F.52	Change Polygon Point command	ECU to VT	182_{10}	$B6_{16}$	Yes	4
F.53	Change Polygon Point response	VT to ECU	182_{10}	$B6_{16}$	No	4
F.54	Change Polygon Scale command	ECU to VT	183_{10}	$B7_{16}$	Yes	4
F.55	Change Polygon Scale response	VT to ECU	183_{10}	$B7_{16}$	No	4
F.56	Graphics Context command	ECU to VT	184_{10}	$B8_{16}$	Yes	4
F.57	Graphics Context response	VT to ECU	184_{10}	$B8_{16}$	No	4
F.58	Get Attribute Value message	ECU to VT	185_{10}	$B9_{16}$	No	4
F.59	Get Attribute Value response	VT to ECU	185_{10}	$B9_{16}$	No	4
F.60	Select Colour Map or Palette command	ECU to VT	186_{10}	BA_{16}	Yes	4
F.61	Select Colour Map or Palette response	VT to ECU	186_{10}	BA_{16}	No	4
F.62	Execute Extended Macro command	ECU to VT	188_{10}	BC_{16}	Yes	5
F.63	Execute Extended Macro response	VT to ECU	188_{10}	BC_{16}	No	5
G.4	Unsupported VT Function message	ECU to VT	253_{10}	FD_{16}	No	5
G.5	VT Unsupported VT Function message	VT to ECU	253_{10}	FD_{16}	No	5
F.64	Select Active Working Set command	ECU to VT	144_{10}	90_{16}	Yes	6
F.65	Select Active Working Set response	VT to ECU	144_{10}	90_{16}	No	6
G.2	VT Status message	VT to ECU	254_{10}	FE_{16}	No	2
G.3	Working Set Maintenance message	ECU to VT	255_{10}	FF_{16}	No	2
H.2	Soft Key Activation message	VT to ECU	0_{10}	00_{16}	No	2
H.3	Soft Key Activation response	ECU to VT	0_{10}	00_{16}	No	2
H.4	Button Activation message	VT to ECU	1_{10}	01_{16}	No	2
H.5	Button Activation response	ECU to VT	1_{10}	01_{16}	No	2

[a] Reserved for future use.

[b] Proprietary commands should not be used between ECUs and VT with different manufacturer codes.

[c] Identifies the VT version at which the command was introduced, however some messages have been revised with later versions.

Table A.5 *(continued)*

Clause/ subclause	Message	PGN	Function (Decimal)	Function (Hex)	Allowed in macro	VT version[c]
H.6	Pointing Event message	VT to ECU	2_{10}	02_{16}	No	2
H.7	Pointing Event response	ECU to VT	2_{10}	02_{16}	No	2
H.8	VT Select Input Object message	VT to ECU	3_{10}	03_{16}	No	2
H.9	VT Select Input Object response	ECU to VT	3_{10}	03_{16}	No	2
H.10	VT ESC message	VT to ECU	4_{10}	04_{16}	No	2
H.11	VT ESC response	ECU to VT	4_{10}	04_{16}	No	2
H.12	VT Change Numeric Value message	VT to ECU	5_{10}	05_{16}	No	2
H.13	VT Change Numeric Value response	ECU to VT	5_{10}	05_{16}	No	2
H.14	VT Change Active Mask message	VT to ECU	6_{10}	06_{16}	No	2
H.15	VT Change Active Mask response	ECU to VT	6_{10}	06_{16}	No	2
H.16	VT Change Soft Key Mask message	VT to ECU	7_{10}	07_{16}	No	2
H.17	VT Change Soft Key Mask response	ECU to VT	7_{10}	07_{16}	No	2
H.18	VT Change String Value message	VT to ECU	8_{10}	08_{16}	No	2
H.19	VT Change String Value response	ECU to VT	8_{10}	08_{16}	No	2
H.20	VT On User-Layout Hide/Show message	VT to ECU	9_{10}	09_{16}	No	4
H.21	VT On User-Layout Hide/Show response	ECU to VT	9_{10}	09_{16}	No	4
H.22	VT Control Audio Signal Termination message	VT to ECU	10_{10}	$0A_{16}$	No	4
J.7.2	Auxiliary Assignment Type 1 command	VT to ECU	32_{10}	20_{16}	No	2
J.7.3	Auxiliary Assignment Type 1 response	ECU to VT	32_{10}	20_{16}	No	2
J.7.4	Auxiliary Input Type 1 status	VT to ECU	33_{10}	21_{16}	No	2
J.7.5	Auxiliary Assignment Type 2 command	VT to ECU	36_{10}	24_{16}	No	3
J.7.6	Auxiliary Assignment Type 2 response	ECU to VT	36_{10}	24_{16}	No	3
J.7.7	Preferred Assignment command	ECU to VT	34_{10}	22_{16}	No	3
J.7.8	Preferred Assignment response	VT to ECU	34_{10}	22_{16}	No	3
J.7.9	Auxiliary Input Type 2 Status message	ECU to VT, VT to ECU	38_{10}	26_{16}	No	3
J.7.10	Auxiliary Input Type 2 Maintenance message	ECU to VT	35_{10}	23_{16}	No	3
J.7.11	Auxiliary Input Status Type 2 Enable command	VT to ECU	37_{10}	25_{16}	No	3
J.7.12	Auxiliary Input Status Type 2 Enable response	ECU to VT	37_{10}	25_{16}	No	3
J.7.13	Auxiliary Capabilities request	ECU to VT	39_{10}	27_{16}	No	5
J.7.14	Auxiliary Capabilities response	VT to ECU	39_{10}	27_{16}	No	5

[a] Reserved for future use.

[b] Proprietary commands should not be used between ECUs and VT with different manufacturer codes.

[c] Identifies the VT version at which the command was introduced, however some messages have been revised with later versions.

Table A.5 *(continued)*

Clause/ subclause	Message	PGN	Function (Decimal)	Function (Hex)	Allowed in macro	VT version[c]
Special Functions						
	Reserved[a]	Any	11_{10}—16_{10}	$0B_{16}$—10_{16}		
	Reserved[a]	Any	19_{10}—31_{10}	13_{16}—$1F_{16}$		
	Reserved[a]	Any	40_{10}—95_{10}	28_{16}—$5F_{16}$		
	Reserved[a]	Any	128_{10}—143_{10}	80_{16}—$8F_{16}$		
	Reserved[a]	Any	145_{10}	91_{16}		
	Reserved[a]	Any	147_{10}—159_{10}	93_{16}—$9F_{16}$		
	Reserved[a]	Any	191_{10}	BF_{16}		
	Reserved[a]	Any	200_{10}—207_{10}	$C8_{16}$—CF_{16}		
	Reserved[a]	Any	211_{10}—222_{10}	$D3_{16}$—DE_{16}		
	Reserved[a]	Any	225_{10}—252_{10}	$E1_{16}$—FC_{16}		
	Proprietary Command[b]	Any	96_{10}—127_{10}	60_{16}—$7F_{16}$		

a Reserved for future use.

b Proprietary commands should not be used between ECUs and VT with different manufacturer codes.

c Identifies the VT version at which the command was introduced, however some messages have been revised with later versions.

Annex B
(normative)

Object definitions

B.1 Working Set object

This object describes a Working Set. Each Working Set shall provide one, and only one, of this object in its object pool. This object shall include one or more objects that fit inside a Soft Key designator for use as an identification of the Working Set. The VT can optionally use the identifier in communication alarms, Auxiliary Control setup, in Soft Keys and any other place where an identifier for the Working Set is required. Only the VT can activate this object. When this object is activated, the associated Working Set "owns" the VT (see Table B.1 and Table B.2).

Allowed commands:

— Change Active Mask command;

— Change Background Colour command;

— Change Child Location command;

— Change Child Position command;

— Get Attribute Value message.

Table B.1 — Working Set events

Event	Caused by	VT behaviour	Message
On Activate	Operator selection of this Working Set via the VT	Deactivate event on current Working Set object. Show event on the active Data Mask of this Working Set object (assuming no alarms).	VT Status message
On Deactivate	Operator selection of a different Working Set via the VT	Hide event on active Data Mask of this Working Set.	VT Status message
On Change Active Mask	Change Active Mask command	Change the active mask attribute. If this Working Set is active, then perform a hide event on the current active mask and a show event on the new active mask.	Change Active Mask response. VT Status message if the Active Mask changed.
On Change Background Colour	Change Background Colour command	If the Working Set designator is visible, fill area with background colour and draw child objects in the order they are listed.	Change Background Colour response
On Change Child Location	Change Child Location command	If the Working Set designator is visible, draw child object at current location in background colour to erase it. Refresh Working Set designator (to redraw child object or objects).	Change Child Location response
On Change Child Position	Change Child Position command	If the Working Set designator is visible, draw child object at current location in background colour to erase it. Refresh Working Set designator (to redraw child object or objects).	Change Child Position response

Table B.2 — Working Set attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=0	3	Object Type = Working Set
Background colour	[1]	Integer	1	0—255	4	Background colour.
Selectable	[2]	Boolean	1	0 or 1	5	0 = FALSE, 1 = TRUE. Indicates whether or not this Working Set can be selected by the operator. VT version 6 and later supports Alarm Masks even when a Working Set is not operator selectable (see 4.6.14). VT version 5 and prior did not provide a clear definition for supporting Alarm Masks when selectable = 0.
Active mask	[3]	Integer	2	0—65534	6—7	The Object ID of the Data or Alarm Mask to display whenever the Working Set is active (or visible, in the case of a VT that supports multiple Working Sets on screen simultaneously). For WS compatible with VT version 6 and later, this value shall reference a valid mask even if not selectable. For WS compatible with VT version 5 and prior, and if the selectable attribute is 0, this attribute is ignored (not range validated).
Number of objects to follow		Integer	1	1—255	8	The objects that follow are used as the Working Set identifier. Although the identifier can be used anywhere, the set of objects shall fit inside a Soft Key designator. The VT clips anything located outside the area of a Soft Key designator.
Number of macros to follow		Integer	1	0—255	9	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Number of languages to follow		Integer	1	0—255	10	The number of language codes to follow. Each two-letter code represents a language that the Working Set can support.
Repeat: {Object ID}		Integer	2	0—65534	11 + object*6	Object ID of an object contained in the designator (see A.1.3). List all objects before listing macros.
{X Location}		Signed integer	2	−32768 to +32767	13 + object*6	Relative X location of the top left corner of the object in VT pixels (relative to the top left corner of the Working Set object).
{Y Location}		Signed integer	2	−32768 to +32767	15 + object*6	Relative Y location of the top left corner of the object in VT pixels (relative to the top left corner of the Working Set object).

Table B.2 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Repeat: {Event ID}		Integer	1	0—255	11 + (No. objects*6)…	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3)
{Macro ID}		Integer	1	0—255	12 + (No. objects*6)…	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)
Repeat: {Language Code}		String	2		Record Position Depends on above	(List these after all objects and macros have been listed.) Two-letter code of a supported language. For language codes, see ISO 639, however all combinations of a-z and A-Z shall be accepted to guard against changes to ISO 639.

B.2 Data Mask object

The Data Mask describes the objects that will appear in the Data Mask area of the physical display. The size of the Data Mask is defined by the VT and accessible to the Working Set with the Get Hardware message (see Table B.3 and Table B.4).

Allowed commands:

— Change Background Colour command;

— Change Child Location command;

— Change Child Position command;

— Change Soft Key Mask command;

— Change Attribute command;

— Get Attribute Value message.

Table B.3 — Data Mask events

Event	Caused by	VT behaviour	Message
On Show	The Data Mask becoming visible on the display.	Fill area with background colour. Draw child objects in the order they are listed in the Data Mask object. Show the associated Soft Key Mask.	VT Status message
On Hide	The Data Mask being removed from the display.	Hide associated Soft Key Mask of this Data Mask.	—
On Refresh	Any action that causes a visible change on a child, grandchild, object, etc.	Redraw objects in the Data Mask that have become corrupted.	—
On Change Background Colour	Change Background Colour command	If the Data Mask is visible, fill area with background colour and draw child objects in the order they are listed.	Change Background Colour response
On Change Child Location	Change Child Location command	Draw child object at current location in background colour to erase it. Refresh Data Mask (to redraw child object or objects).	Change Child Location response
On Change Child Position	Change Child Position command	Draw child object at current location in background colour to erase it. Refresh Data Mask (to redraw child object or objects).	Change Child Position response
On Change Soft Key Mask	Change Soft Key Mask command	If the Data Mask is visible, hide event on the current Soft Key Mask and a show event on the new Soft Key Mask.	Change Soft Key Mask response. If this mask is visible, VT Status message.
On Change Attribute	Change Attribute command	See Change Background Colour command and Change Soft Key Mask command behaviour above.	Change Attribute response. If change affects visible masks, VT Status message.
On Pointing Event press	Operator touches data mask	—	Pointing Event message
On Pointing Event release	Operator touch is released	—	Pointing Event message

Table B.4 — Data mask attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=1	3	Object Type = Data mask
Background colour	1	Integer	1	0—255	4	Background colour.
Soft Key Mask	2	Integer	2	0—65534, 65535	5—6	Object ID of a Soft Key Mask associated with this Data Mask. Whenever this Data Mask is displayed, the associated Soft Key Mask is also displayed. If the NULL Object ID is used, there are no Soft Keys associated with this Data Mask and the Soft Key designators should be cleared.
Number of objects to follow		Integer	1	0—255	7	Number of objects to follow even if zero. Each of these objects is “contained” in this Data Mask. Each object consists of 6 bytes: two (2) for Object ID and four (4) for location.

Table B.4 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Number of macros to follow		Integer	1	0—255	8	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Object ID}		Integer	2	0—65534	9 + object*6	Object ID of an object contained in this mask (see A.1.3). List all objects before listing macros.
{X Location}		Signed integer	2	−32768 to +32767	11 + object*6	Relative X location of the top left corner of the object (relative to the top left corner of the Data Mask).
{Y Location}		Signed integer	2	−32768 to +32767	13 + object*6	Relative Y location of the top left corner of the object (relative to the top left corner of the Data Mask).
Repeat: {Event ID}		Integer	1	0—255	9 + (No. objects*6)	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3)
{Macro ID}		Integer	1	0—255	10 + (No. objects*6)	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)

B.3 Alarm Mask object

For information on Alarm Mask behaviour, see 4.6.14, Table B.5 and Table B.6.

Allowed commands:

— Change Background Colour command;

— Change Child Location command;

— Change Child Position command;

— Change Priority command;

— Change Soft Key Mask command;

— Change Attribute command;

— Get Attribute Value message.

Table B.5 — Alarm Mask events

Event	Caused by	VT behaviour	Message
On Show	The Alarm Mask becoming visible on the display.	Fill area with background colour. Draw child objects in the order they are listed in the Alarm Mask object. Show the associated Soft Key Mask.	VT Status message
On Hide	The mask being removed from the display.	Hide associated Soft Key Mask of this Alarm Mask.	—
On Refresh	Any action that causes a show or hide on a child, grandchild, etc. object.	Redraw objects in the Alarm Mask that have become corrupted.	—
On Change Background Colour	Change Background Colour command	If the Alarm Mask is visible, fill area with background colour and draw child objects in the order they are listed.	Change Background Colour response
On Change Child Location	Change Child Location command	Draw child object at current location in background colour to erase it. Refresh Alarm Mask (to redraw child object or objects).	Change Child Location response
On Change Child Position	Change Child Position command	Draw child object at current location in background colour to erase it. Refresh Alarm Mask (to redraw child object or objects).	Change Child Position response
On Change Priority	Change Priority command	If this is the current mask in this Working Set then reevaluate alarm priorities as follows: a) If this Alarm Mask is visible and it is no longer the highest priority alarm then deactivate this Working Set and activate the WS with the highest priority alarm b) If this Alarm Mask is not visible and it becomes the highest priority alarm then deactivate the current Working Set and activate this WS.	Change Priority response
On Change Soft Key Mask	Change Soft Key Mask command	If the Alarm Mask is visible, hide event on the current Soft Key Mask and a show event on the new Soft Key Mask.	Change Soft Key Mask response. If this mask is visible, VT Status message.
On Change Attribute	Change Attribute command	For behaviour see other change commands above.	Change Attribute response. If change affects visible masks, VT Status message.

Table B.6 — Alarm Mask attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=2	3	Object Type = Alarm Mask
Background colour	1	Integer	1	0—255	4	Background colour.

Table B.6 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Soft Key Mask	2	Integer	2	0—65534, 65535	5—6	Object ID of a Soft Key Mask associated with this Alarm Mask. Whenever this Alarm Mask is displayed, the associated Soft Key Mask is also displayed. If the NULL is used, there are no Soft Keys associated with this Alarm Mask and the Soft Key designators should be cleared.
Priority	3	Integer	1	0—2	7	Priority of this alarm as follows: 0 = High, operator is in danger or urgent machine malfunction 1 = Medium, normal alarm, machine is malfunctioning 2 = Low, information only
Acoustic signal	4	Integer	1	0—3	8	Acoustic signal. 0 = highest priority, 1 = medium priority, 2 = lowest priority, 3 = none (silent).
Number of objects to follow		Integer	1	0—255	9	Number of objects to follow even if zero. Each of these objects is "contained" in this Alarm Mask. Each object consists of 6 bytes: two (2) for Object ID and four (4) for location.
Number of macros to follow		Integer	1	0—255	10	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Object ID}		Integer	2	0—65534	11 + object*6	Object ID of an object contained in this mask (see A.1.3). List all objects before listing macros.
{X Location}		Signed integer	2	−32768 to +32767	13 + object*6	Relative X location of the top left corner of the object (relative to the top left corner of the Alarm Mask).

Table B.6 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
{Y Location}		Signed integer	2	−32768 to +32767	15 + object*6	Relative Y location of the top left corner of the object (relative to the top left corner of the Alarm Mask).
Repeat: {Event ID}		Integer	1	0—255	11 + (No. objects*6)...	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3)
{Macro ID}		Integer	1	0—255	12 + (No. objects*6)...	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)

B.4 Container object

The Container object is used to group objects for the purpose of moving, hiding or sharing the group. A container is not a visible object, only a logical grouping of other objects. Unlike masks, containers can be hidden and shown at run-time under Working Set control. The Container object has defined size limits to assist in determining when other objects are overlaid with the container. See Table B.7 and Table B.8.

Allowed commands:

— Hide/Show Object command;

— Change Child Location command;

— Change Child Position command;

— Change Size command;

— Get Attribute Value message.

Table B.7 — Container events

Event	Caused by	VT behaviour	Message
On Show	Show command on this object even if the container is already visible.	Draw the contained objects in the order listed. Refresh parent mask.	Hide/Show Response but only if triggered by Hide/Show command
On Hide	Hide command on this object even if the container is already hidden.	Redraw the object with the mask's background colour. Refresh parent mask.	Hide/Show Response but only if triggered by Hide/Show command
On Refresh	Any action that causes a show or hide on a child, grandchild etc. object.	Redraw objects in the container that have become corrupted.	—
On Change Child Location	Change Child Location command	Draw child object at current location in background colour to erase it. Refresh container (to redraw child object or objects).	Change Child Location response
On Change Child Position	Change Child Position command	Draw child object at current location in background colour to erase it. Refresh container (to redraw child object or objects).	Change Child Position response
On Change Size	Change Size command	Draw child objects at current location in background colour to erase them. Refresh container (to redraw child object or objects).	Change Size response

Table B.8 — Container attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=3	3	Object Type = Container
Width	[1]	Integer	2	0—65535	4—5	Maximum width of the container's area in pixels. Objects or portions of objects outside the defined area are clipped.
Height	[2]	Integer	2	0—65535	6—7	Maximum height of the container's area in pixels. Objects or portions of objects outside the defined area are clipped.
Hidden	[3]	Boolean	1	0 or 1	8	0 = FALSE, 1 = TRUE. Indicates whether or not this container and its child objects are hidden (not displayed). (TRUE = Hidden)
Number of objects to follow		Integer	1	0—255	9	Number of objects to follow even if zero. Each of these objects is "contained" in this object. Each object consists of 6 bytes: two (2) for Object ID and four (4) for location.
Number of macros to follow		Integer	1	0—255	10	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.

Table B.8 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Repeat: {Object ID}		Integer	2	0—65534	11 + object*6	Object ID of an object contained in this container (see A.1.3). List all objects before listing macros.
{X Location}		Signed integer	2	−32768 to +32767	13 + object*6	Relative X location of the top left corner of the object (relative to the top left corner of the Container object).
{Y Location}		Signed integer	2	−32768 to +32767	15 + object*6	Relative Y location of the top left corner of the object (relative to the top left corner of the Container object).
Repeat: {Event ID}		Integer	1	0—255	11 + (No. objects*6)...	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3)
{Macro ID}		Integer	1	0—255	12 + (No. objects*6)...	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)

B.5 Soft Key Mask object

The Soft Key Mask is a Container object that contains Key objects, Object Pointer objects, or External Object Pointer objects. The pointer objects (see 4.6.11.5, 4.6.11.6) can only resolve to a NULL object or a Key object. Keys are assigned to physical Soft Keys in the order listed. It is allowable for a Soft Key Mask to contain no Keys in order that all Soft Keys are effectively disabled when this mask is activated. See Table B.9 and Table B.10.

A common implementation, which was not clearly defined until VT version 4 and later, is that pointers to the NULL Object ID reserve a Soft Key position (the remaining Soft Keys do not move up and the trailing Soft Keys can be navigated to). Pointers to NULL that are at the end of the list of Soft Keys shall not be displayed. They should not be considered for paging. The paging requirements can be dynamic at run-time based if pointer values are changed to and from NULL (see 4.5.3.5).

Allowed commands:

— Change Background Colour command;

— Change Attribute command;

— Get Attribute Value message.

Table B.9 — Soft Key Mask events

Event	Caused by	VT behaviour	Message
On Show	The mask becoming visible on the display.	Draw child objects in the order they are listed in the Soft Key Mask.	—
On Hide	The mask being removed from the display.	—	—
On Change Background Colour	Change Background Colour command	If the Soft Key Mask is visible, fill area with background colour and draw child objects in the order they are listed.	Change Background Colour response
On Change Attribute	Change Attribute command	For behaviour, see Change Background Colour command.	Change Attribute response

Table B.10 — Soft Key Mask attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=4	3	Object Type = Soft Key Mask
Background colour	1	Integer	1	0—255	4	Background colour. The Key object has its own background colour attribute that overrides this attribute.
Number of objects to follow		Integer	1	0—255	5	Number of objects to follow even if zero. Each of these objects is “contained” in this Soft Key Mask.
Number of macros to follow		Integer	1	0—255	6	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Object ID}		Integer	2	0—65534	7 + object*2...	Object ID of an object contained in this mask (see A.1.3).
Repeat: {Event ID}		Integer	1	0—255	7 + (No. objects*2)...	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3)
{Macro ID}		Integer	1	0—255	8 + (No. objects*2)...	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)

B.6 Key object

The Key object defines the designator and key code for a Soft Key. Any object located outside of a Soft Key designator is clipped. See Table B.11 and Table B.12.

Allowed commands:

— Select Input Object command (VT version 4 and later);

— Change Background Colour command;

— Change Child Location command;

— Change Child Position command;

— Change Attribute command;

— Get Attribute Value message.

Table B.11 — Key events

Event	Caused by	VT behaviour	Message
On Key Press	Operator pressing the Soft Key	—	Soft Key Activation message
On Key Release	Operator releasing the Soft Key	—	Soft Key Activation message
On Change Background Colour	Change Background Colour command	If the key is visible, fill area with background colour and draw child objects in the order they are listed.	Change Background Colour response
On Change Child Location	Change Child Location command	Draw child object at current location in background colour to erase it. Refresh Key Designator (to redraw child object or objects).	Change Child Location response
On Change Child Position	Change Child Position command	Draw child object at current location in background colour to erase it. Refresh Key Designator (to redraw child object or objects).	Change Child Position response
On Change Attribute	Change Attribute command	For behaviour see other change commands above.	Change Attribute response
On Input Field Selection	Select Input Object command or operator navigates to Key object	The VT shall provide some way for the operator to recognize that the Key object is selected (has focus).	Select Input Object response or VT Select Input Object message
On Input Field De-selection	Select Input Object command or operator navigates off Key object	—	Select Input Object response or VT Select Input Object

Table B.12 — Key attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=5	3	Object type = key
Background colour	1	Integer	1	0—255	4	Background colour.

Table B.12 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Key code	2	Integer	1	1—255	5	VT reports this code in the Soft Key Activation message. NOTE Key code zero (0) is reserved for use for the ACK means.
Number of objects to follow		Integer	1	0—255	6	Number of objects to follow even if zero. Each of these objects is “contained” in this object. Each object consists of 6 bytes: two (2) for object id and four (4) for location.
Number of macros to follow		Integer	1	0—255	7	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Object ID}		Integer	2	0—65534	8 + object*6	Object ID of an object contained in this key (see A.1.3). List all objects before listing macros.
{X Location}		Signed integer	2	− 32768 to +32767	10 + object*6	Relative X location of the top left corner of the object in VT pixels (relative to the top left corner of the Soft Key designator).
{Y Location}		Signed integer	2	−32768 to +32767	12 + object*6	Relative Y location of the top left corner of the object in VT pixels (relative to the top left corner of the Soft Key designator).
Repeat: {Event ID}		Integer	1	0—255	8 + (No. objects*6)	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3)
{Macro ID}		Integer	1	0—255	9 + (No. objects*6)	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)

B.7 Button object

The Button object defines a button control. This object is intended mainly for VTs with touch screens or a pointing method but shall be supported by all VTs. Alternatively, if a touch screen or a pointing method is not supported, the VT shall provide a means for navigating to the Button object (or objects) (see 4.6.17). The Working Set can determine if the VT supports touch screen or pointing devices by using a Get Hardware message. When the Button object is activated, the VT sends a Button Activation message to the Working Set Master.

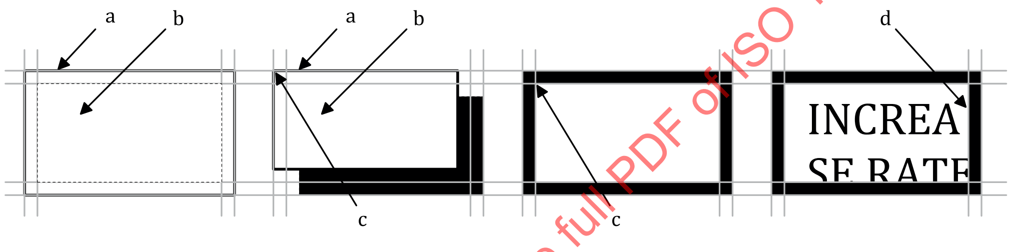
The VT shall indicate when a Button is selected (has focus), pressed or latched depending on the options attribute.

The Button consists of the Button Area, the Button Face, and the Button Border.

— The Button Area is the area which is defined by the Button object width and height attributes.

— The Button Face is the area which contains the Background colour and into which the designer can implement child objects. Child objects are clipped to the width and height of the Button Face [see Figure B.1 (Options — Bit 5 = FALSE)]. The Button Face is 8 pixels smaller (in both width and height) than the Button Area, unless the Options—No border bit is set to TRUE, in which case the Button Face is extended to be equal to the Button Area.

— The Button Border is a VT proprietary 8-pixel area which contains pixels generally rendered from the border colour attribute. Border location and width on any side of the button is VT proprietary [see Figure B.1 (Options — Bit 5 = FALSE)]. Presentation of the border colour characteristics (transparency, hue, saturation and brightness) is VT proprietary, possibly using colours not within the standard colour palette. As a result, the position of the button face, within the bounding rectangle defined by the button width and height, is VT proprietary. The Button Border can be hidden by setting the Options — Suppress border bit to TRUE. The Button Border can be eliminated by setting the Options — No border bit to TRUE. See Table B.13 and Table B.14.



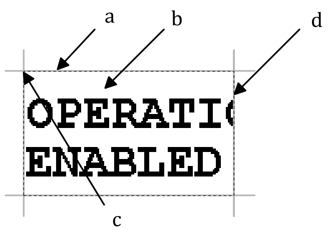
a Button Area is determined by width and height attributes.

b Button Face is 8 pixels smaller in both widths and height.

c Button Face offset is VT proprietary (0,0) ⇐ offset ⇐ (8,8).

d Objects do not fit; the Button Face is clipped.

Figure B.1 — Button examples with border (Options — Bit 5 = FALSE)



a Button Area is determined by width and height attributes.

b Button Face is 8 pixels smaller in both widths and height.

c Button Face offset is (0,0).

d Objects do not fit; the Button Face is clipped.

Figure B.2 — Button examples no border (Options — Bit 5 = TRUE)

Allowed commands:

— Enable/Disable Object command (VT version 4 and later);

— Select Input Object command (VT version 4 and later);

— Change Background Colour command;

— Change Size command;

— Change Child Location command;

— Change Child Position command;

— Change Attribute command;

— Get Attribute Value message.

Table B.13 — Button events

Event	Caused by	VT behaviour	Message
On Enable	Enable/Disable command	Mark the Button object enabled. If displayed, operator can navigate to it.	Enable/Disable Object response
On Disable	Enable/Disable command	Mark the Button object disabled. Even if displayed, the operator cannot select this object. VT shall make it clear to the operator that the Button object is disabled.	Enable/Disable Object response
On Input Field Selection	Select Input Object command or operator navigates to Button object	The VT shall provide some way for the operator to recognize that the Button object is selected (has focus).	Select Input Object response or VT Select Input Object message
On input Field De-selection	Select Input Object command or operator navigates off Button object or as a result of a disable event	—	Select Input Object response or VT Select Input Object
On Key Press	Operator activating the Button or Change Attribute command changing the current Button state for latchable Buttons	—	If Operator activates the Button, send Button Activation
On Key Release	Operator releasing the Button or Change Attribute command changing the current Button state for latchable Buttons	—	If Operator releases the Button, send Button Activation
On Change Background Colour	Change Background Colour command	If the Button is visible, fill area with background colour and draw child objects in the order they are listed.	Change Background Colour response
On Change Size	Change Size command	Draw child objects at current location in background colour to erase them. Refresh parent mask.	Change Size response
On Change Child Location	Change Child Location command	Draw child object at current location in background colour to erase it. Refresh Button (to redraw child object or objects).	Change Child Location response
On Change Child Position	Change Child Position command	Draw child object at current location in background colour to erase it. Refresh Button (to redraw child object or objects).	Change Child Position response
On Change Attribute	Change Attribute command	For behaviour, see other change commands above.	Change Attribute response

Table B.14 — Button attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=6	3	Object Type = Button
Width	1	Integer	2	0—65535	4—5	Maximum width of the Button's area in pixels.
Height	2	Integer	2	0—65535	6—7	Maximum height of the Button's area in pixels.
Background colour	3	Integer	1	0—255	8	Background colour.
Border colour	4	Integer	1	0—255	9	Border colour.
Key Code	5	Integer	1	0—255	10	VT reports this code in the Button Activation message.
Options	6 [c]	Bitmask	1	0,1,3[a] 0—63[b]	11	TRUE (1) or FALSE (0). Bit 0 = If TRUE, the Button is "latchable" and remains pressed until the next activation. If FALSE, the Button is momentary. This bit cannot be changed during runtime by a Change Attribute command. Any change in this bit shall be ignored by the VT. Bit 1 = Current Button state for latchable Buttons. 0 = released, 1 = latched. This attribute is ignored for momentary Buttons. Bit 2 = Suppress border. If FALSE, VT draws the proprietary border. If TRUE, no border is ever drawn (even when pressed momentarily or latched) and the area normally occupied by the border is always transparent.[b]

[a] VT version 3 and prior.

[b] VT version 4 and later

[c] This AID is present in VT version 4 and later.

Table B.14 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
						Bit 3 = Transparent Background. If FALSE, the Button's interior background is filled using the background colour attribute. If TRUE, the Button's background is always transparent and the background colour attribute is not used.[b] Bit 4 = Disabled. If FALSE, the Button is enabled and can be selected and activated by the operator. If TRUE, the Button is drawn disabled (method proprietary to VT) and it cannot be selected or activated by the operator.[b] Bit 5 = No border. If FALSE, the Button Border area is used by the VT as described in Bit 2. If TRUE, Bit 2 is ignored therefore no border is ever drawn (even when pressed momentarily or latched) and the Button Face extends to the full Button Area.[b] Bits 6—7 = reserved, set to 0 NOTE By using a momentary Button, in combination with bits 2, 3, and 5 and by modifying the Button appearance in real time, a Working Set can create "radio button" type behaviour between several Button objects.
Number of objects to follow		Integer	1	0—255	12	Number of objects to follow even if zero. Each of these objects is "contained" in this object. Each object consists of 6 bytes: two (2) for Object ID and four (4) for location.
Number of macros to follow		Integer	1	0—255	13	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Object ID}		Integer	2	0—65534	14 + object*6	Object ID of an object contained in this Button (See A.1.3). List all objects before listing macros.
{X Location}		Signed integer	2	−32768 to +32767	16 + object*6	Relative X location of the top left corner of the object in VT pixels (relative to the top left inner corner of the Button). Objects or portions of objects outside the inner border are clipped.

[a] VT version 3 and prior.

[b] VT version 4 and later

[c] This AID is present in VT version 4 and later.

Table B.14 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
{Y Location}		Signed integer	2	−32768 to +32767	18 + object*6	Relative Y location of the top left corner of the object in VT pixels (relative to the top left inner corner of the Button). Objects or portions of objects outside the inner border are clipped.
Repeat: {Event ID}		Integer	1	0—255	14 + (No. objects*6)	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3)
{Macro ID}		Integer	1	0—255	15 + (No. objects*6)	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)

[a] VT version 3 and prior.

[b] VT version 4 and later

[c] This AID is present in VT version 4 and later.

B.8 Input field objects

B.8.1 General

There are four types of input field: Boolean, String, Number, and List. No border shall be drawn around any input field by the VT.

Input Boolean, Input String object, Input Number and Input List objects have similar relationships, commands and events and they are listed here. The attributes for each object differ.

Input objects have three states:

a) hidden (not displayed);

b) shown and enabled (displayed and able to accept input);

c) shown and disabled (displayed but not able to accept input).

Input field objects do not have a show/hide attribute. In order to hide an input field object, it can be contained by a Container object or Object Pointer object. See Table B.15 to Table B.20.

Allowed Commands (see B.8.5):

— Enable/Disable Object command;

— Select Input Object command;

— ESC command;

— Change Background Colour command (excluding Input List object);

— Change Numeric Value command (excluding Input String object);

— Change String Value command (Input String object only);

— Change Attribute command;

— Change Size command;

— Get Attribute Value message.

Table B.15 — Input events

Event	Caused by	VT Behaviour	Message
On Refresh	See Data Mask refresh for caused by conditions	Redraw this object.	—
On Enable	Enable/Disable Object command	Mark the input object enabled. If displayed, operator can navigate to it.	Enable/Disable Object Response
On Disable	Enable/Disable Object command	Mark the input object disabled. Even if displayed, the operator cannot select this object for input. VT shall make it clear to the operator that the input field is disabled.	Enable/Disable Object Response
On Input Field Selection	Select Input Object command or operator navigates to input field	The VT shall provide some way for the operator to recognize that the input field is selected (has focus).	Select Input Object response or VT Select Input Object message
On Input Field De-selection	Select Input Object command or operator navigates off input field or as a result of a disable event	—	Select Input Object response or VT Select Input Object message
On ESC	Operator aborts input using ESC key or Working Set sends an ESC command	If the input object is not enabled for real time editing, then revert the value of the object to the value before operator began the input. (See 4.2, data input, real time editing). Redraw this object. Refresh parent object.	VT ESC message or ESC response (see 4.6.17).
On Change Background Colour	Change Background Colour command	If the input field is visible, fill area with background colour and redraw the object with the new background colour.	Change Background Colour Response
On Change Value	Change Numeric Value command or Change String Value command	If input object is displayed, redraw object with new value. Refresh parent object.	Change Numeric Value response or Change String Value response
On Entry of Value	Operator saving changes by use of the ENTER means regardless of whether or not the value changed	VT updates the Working Set with new value.	VT Change Numeric Value message or VT Change String Value message

Table B.15 *(continued)*

Event	Caused by	VT Behaviour	Message
On Entry of New Value	Operator saving changes by use of the ENTER means when value has changed	Working Set is notified by the "On Entry of value" event so no additional notification is required on this event.	—
On Change Attribute	Change Attribute command	If field is visible, refresh.	Change Attribute response.
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size response

B.8.2 Input Boolean object

The Input Boolean object is used to input a TRUE/FALSE type indication from the operator. This is a graphical object and the appearance of the indicator when the value is > 0 is left to the VT, but it shall fit in the square area specified by the width attribute. An example of a Boolean input is a checkbox.

When the value is 0, the object area is the background colour.

When the value is > 0, the VT draws the indicator using the foreground colour on the background colour.

NOTE VT Version 3 and prior have a Value range in the set {0, 1} but do not clarify the presentation when the value is not in the set {0, 1}, which is possible when using a variable reference. In VT version 4 and later, the TRUE indication is shown for any value > 0, and the FALSE indication for the value = 0. When the Input Boolean value is changed, the VT indicates the state of the Input Boolean to the Working Set with a value in the set {0, 1}.

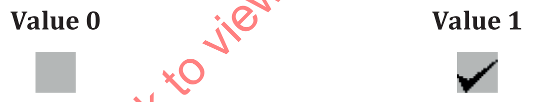


Figure B.3 — Input Boolean examples

Table B.16 — Input Boolean attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=7	3	Object Type = Input Boolean
Background colour	1	Integer	1	0—255	4	Background colour.
Width	2	Integer	2	0—65535	5—6	Maximum width and height of the input field in pixels.
Foreground colour	3	Integer	2	0—65534	7—8	Object ID of a Font Attributes object to use for display formatting of this field. The only useful attribute is the font colour
Variable reference	4	Integer	2	0—65534, 65535	9—10	Object ID of a Number Variable object in which to store or retrieve the object's value. If this attribute is set to NULL, the value is stored directly in the value attribute instead.

[a] VT version 4 and later.

Table B.16 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Value	[5]	Integer	1	0, 1—255	11	Value of the input field. 0 for FALSE or > 0 for TRUE. Used only if variable reference attribute is NULL.
Enabled	[6][a]	Integer	1	0 or 1	12	Current state of object. 0 = Disabled, 1 = Enabled
Number of macros to follow		Integer	1	0—255	13	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0—255	14…	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3)
{Macro ID}		Integer	1	0—255	15…	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)

[a] VT version 4 and later.

B.8.3 Input String object

This object is used to input a character string from the operator. Displayable characters are shown in Table K.2 to Table K.8. Several special formatting characters are permitted in the Input String object value and shall be properly interpreted by the VT, as specified (see 4.6.19.6).

Table B.17 — Input String attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=8	3	Object Type = Input String object
Width	1	Integer	2	0—65535	4—5	Maximum width of the input field in pixels. Objects or portions of objects outside the defined area are clipped.

[a] VT version 3 and prior.

[b] VT version 4 and later.

[c] For VT version 3 and prior, if the Variable reference is not NULL, the Length attribute shall be set to 0.

Table B.17 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Height	2	Integer	2	0—65535	6—7	Maximum height of the input field in pixels. Objects or portions of objects outside the defined area are clipped.
Background colour	3	Integer	1	0—255	8	Background colour. Used only if the "transparent" bit in the options attribute is cleared. To be applied in the complete rectangle specified by Width and Height.
Font attributes	4	Integer	2	0—65534	9—10	Object ID of a Font Attributes object to use for display formatting of this field.
Input attributes	5	Integer	2	0—65534, 65535	11—12	Object ID of an Input Attributes object or Extended Input Attributes object to use for character string validation or NULL for no validation. A referenced Input Attributes object or or Extended Input Attributes object shall be of the same type as the string value (or String Variable), in that both shall be either an 8-bit string, or a WideString. If they are not of the same type no validation shall be performed. The indicated object and the Value match if the indicated object is an Input Attributes object and the Value is an 8-bit string, or if the indicated object is an or Extended Input Attributes object and the Value is a WideString.
Options	6	Bitmask	1	0—3[a] 0—7[b]	13	Logical bits to indicate options. 1 = TRUE. Bit 0 = Transparent. If TRUE, the input field is displayed with background showing through instead of using the background colour attribute. Bit 1 = Auto-Wrap. If TRUE, Auto-Wrapping rules apply (see 4.6.19.5) Bit 2 = Wrap on Hyphen. If TRUE, Auto-Wrapping can occur between a hyphen ($2D_{16}$) and the following character (see 4.6.19.5). Wrap on Hyphen is a modifier to the Auto-Wrap option and is applied only if the Auto-Wrap option is TRUE and ignored if the Auto-Wrap option is FALSE. [b]
Variable reference	7	Integer	2	0—65534, 65535	14—15	Object ID of a String Variable object in which to store or retrieve the object's value. If this attribute is set to NULL, the string is stored directly in the value attribute instead. If this attribute is not NULL (≤ 65534) the Length and Value attributes are not used.[c]

[a] VT version 3 and prior.

[b] VT version 4 and later.

[c] For VT version 3 and prior, if the Variable reference is not NULL, the Length attribute shall be set to 0.

Table B.17 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Justification	8	Integer	1	0—2[a] 0—15[b]	16	Field justification. Indicates how the text string is positioned within the field defined by width and height. See 4.6.19.1. Horizontal Justification Value of Bits 0 — 1 0 = Position Left 1 = Position Middle 2 = Position Right 3 = Reserved Vertical Justification Value of Bits 2 — 3[b] 0 = Position Top 1 = Position Middle 2 = Position Bottom 3 = Reserved During data input of this object, the VT designer can choose to suppress justification until the field is closed after input
Length		Integer	1	0—255	17	Maximum fixed length of the Input String object value in bytes. This can be set to 0 if a variable reference is used. When variable reference is used, its variable shall not exceed 255 bytes, since the length attribute of the Input String object Value command (H.18) is only one byte.
Value		String	Length		18...	Value of the input field. Used only if variable reference attribute is NULL. This attribute shall have the size indicated by the Length attribute — even if a variable reference is used. Pad with spaces as necessary to satisfy length attribute. The text can be either 8-bit or WideString (see 4.6.19.7). The string type (8-bit/WideString) shall not be changed by the VT, however the Working Set can cause the type to change from an 8-bit String to a WideString or vice versa.
Enabled	[9][b]	Integer	1	0 or 1	Depends on size of value attribute	Current state of object. 0 = Disabled, 1 = Enabled

[a] VT version 3 and prior.

[b] VT version 4 and later.

[c] For VT version 3 and prior, if the Variable reference is not NULL, the Length attribute shall be set to 0.

Table B.17 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Number of macros to follow		Integer	1	0—255	Depends on size of value attribute	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0—255	Depends on size of value attribute	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3)
{Macro ID}		Integer	1	0—255	Depends on size of value attribute	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)

[a] VT version 3 and prior.

[b] VT version 4 and later.

[C] For VT version 3 and prior, if the Variable reference is not NULL, the Length attribute shall be set to 0.

B.8.4 Input Number object

This object is used to format, display and change a numeric value based on a supplied integer value. The VT shall use the following equation to format the displayed value, even if the value is outside the min/max range:

Displayed value = (value attribute + Offset) × Scaling Factor

Depending on the "Options" attribute in Table B.18, displayed values are either truncated or rounded to the number of decimals specified in the "Number of decimals" attribute.

The VT should implement double precision operations to minimize rounding errors.

When the operator presses the Enter means, to close the input object after data input, the VT shall only accept the new value if the following equations are true:

Scaled max value = (Max value + Offset) × Scaling Factor

Scaled min value = (Min value + Offset) × Scaling Factor

Scaled min value ≤ new value ≤ Scaled max value

If the above equations are not true the VT shall ignore the Enter means and keep the input object open for data input.

If the above equations are true the VT sets the value attribute of the input number object or the referenced numeric value object according to the following equation:

Value attribute = (new value/Scaling Factor) — Offset

NOTE While the operator is not allowed to enter values outside the min/max range, the Working Set is allowed to set any value either by pool upload or by the Change Numeric Value command.

Table B.18 — Input Number attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=9	3	Object Type = Input Number
Width	1	Integer	2	0—65535	4—5	Maximum width of the input field in pixels. Objects or portions of objects outside the defined area are clipped.
Height	2	Integer	2	0—65535	6—7	Maximum height of the input field in pixels. Objects or portions of objects outside the defined area are clipped.
Background colour	3	Integer	1	0—255	8	Background colour. Used only if the "transparent" bit in the options attribute is cleared. To be applied in the complete rectangle specified by Width and Height.
Font attributes	4	Integer	2	0—65534	9—10	Object ID of a Font Attributes object to use for display formatting of this field.
Options	5	Bitmask	1	0—7[a] 0—15[b]	11	Logical bits to indicate options. 1 = TRUE. Bit 0 = Transparent. If TRUE, the input field is displayed with background showing through instead of using the background colour attribute. Bit 1 = Display leading zeros. If TRUE, fill left to width of field with zeros; justification is applied after filling to the width of the field with zeros. Bit 2 = Display zero as blank if this bit is TRUE. When this option bit is set, a blank field is displayed if and only if the displayed value of the object is exactly zero. Except when the field is blank, the VT shall always display at least one digit before the decimal point. (examples: 2,2, 0,2) Bit 3 = Truncate. If TRUE the value shall be truncated to the specified number of decimals. Otherwise it shall be rounded off to the specified number of decimals.[b, c] Designer should account for a unary minus sign with respect to leading zeros and the field width.
Variable reference	6	Integer	2	0—65534, 65535	12—13	Object ID of a Number Variable object in which to store or retrieve the object's raw unscaled value. If this attribute is set to NULL, the value is stored directly in the value attribute instead. VT transmits the raw unscaled value to the Working Set.

[a] VT version 3 and prior.

[b] VT Version 4 and later.

[c] Prior to VT version 4, the behaviour was undefined.

Table B.18 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Value	[14]	Integer	4	0 to 2^32—1	14—17	Raw unsigned value of the input field before scaling (unsigned 32-bit integer). Used only if variable reference attribute is NULL. VT transmits the raw unscaled value to the Working Set.
Min value	7	Integer	4	0 to 2^32—1	18—21	Raw minimum value for the input before scaling. Offset and scaling shall be applied to determine the actual minimum value.
Max value	8	Integer	4	0 to 2^32—1	22—25	Raw maximum value for the input. Offset and scaling shall be applied to determine the actual maximum value.
Offset	9	Signed Integer	4	−2^31 to 2^31—1	26—29	Offset to be applied to the input value and min/max values (32-bit signed integer).
Scale	10	Float	4		30—33	Scale to be applied to the input value and min/max values.
Number of decimals	11	Integer	1	0—7	34	Specifies number of decimals to display after the decimal point.
Format	12	Boolean	1	0 or 1	35	0 = use fixed format decimal display (####.nn) 1 = use exponential format ([-]###.nnE[±]##) where n is set by the number of decimals attribute.
Justification	13	Integer	1	0—2[a] 0—15[b]	36	Field justification. Indicates how the number is positioned within the field defined by width and height. See 4.6.19.1. Horizontal Justification Value of Bits 0 — 1 0 = Position Left 1 = Position Middle 2 = Position Right 3 = Reserved Vertical Justification Value of Bits 2 — 3[b] 0 = Position Top 1 = Position Middle 2 = Position Bottom 3 = Reserved During data input of this object, the VT designer can choose to suppress justification until the field is closed after input .
Options 2	[15][b]	Integer	1	0, 1[a] 0 — 3[b]	37	Logical bits to indicate options. 1 = TRUE. Bit 0 = Enabled. If TRUE the object shall be enabled. If FALSE, the object is disabled. Bit 1 = real time editing. [b] If TRUE the value shall be transmitted to the Working Set Master as it is being changed (see Real Time in Clause 4.2).

[a] VT version 3 and prior.

[b] VT Version 4 and later.

[c] Prior to VT version 4, the behaviour was undefined.

Table B.18 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Number of macros to follow		Integer	1	0–255	38	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0–255	39...	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3)
{Macro ID}		Integer	1	0–255	40...	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)

[a] VT version 3 and prior.

[b] VT Version 4 and later.

[c] Prior to VT version 4, the behaviour was undefined.

B.8.5 Input List object

The Input List object is used to show one object out of a set of objects, and to allow operator selection of one object from the set. The object to show is determined by the Value attribute or a Variable reference.

The exact implementation and appearance of the Input List object is proprietary to the VT. For example, a simple implementation of the Input List object could be to display values as the operator moves through the list using +/− keys. A more complex implementation could be to draw a graphical pop-up list box with a scroll bar for moving through the allowable values. In either case, only the current value shall be displayed when this object is not open for edit. The width and height attributes define the width and height of the displayed value only.

This object is used to select an item from a list of objects. The value transmitted to the Working Set Master is the list index chosen (range 0-254).

NOTE In VT version 3 and prior, the behaviour with the value 255 was not defined. The value 255 is used to indicate that no item is chosen. The CF can set the value to 255 for this purpose.

The operator, in the process of selecting a list item, shall not be allowed to set the value to 255.

The operator also shall not be allowed to set the value to an invalid index (an index which is greater than the number of items in the list minus 1). However, if the list references a number variable, then it is possible for that number variable to be set by the Working Set or by the operator (by using a different input object) to a value which is not a valid index for the list.

When a list item is an Object Pointer with a value of NULL or is a Container, and the Container is in the hidden state, it is considered an empty object. It will still occupy a position in the displayed list, so it can still be selected by the operator, even though its contents will not be visible.

When a list item has an Object ID of NULL it is considered an invisible object. It does not occupy a position in the displayed list and cannot be selected by the operator. However, it is still counted when determining list indexes and maintains its position in the list even though it is not visible to the operator.

The VT shall not display anything for the selected item in the following cases:

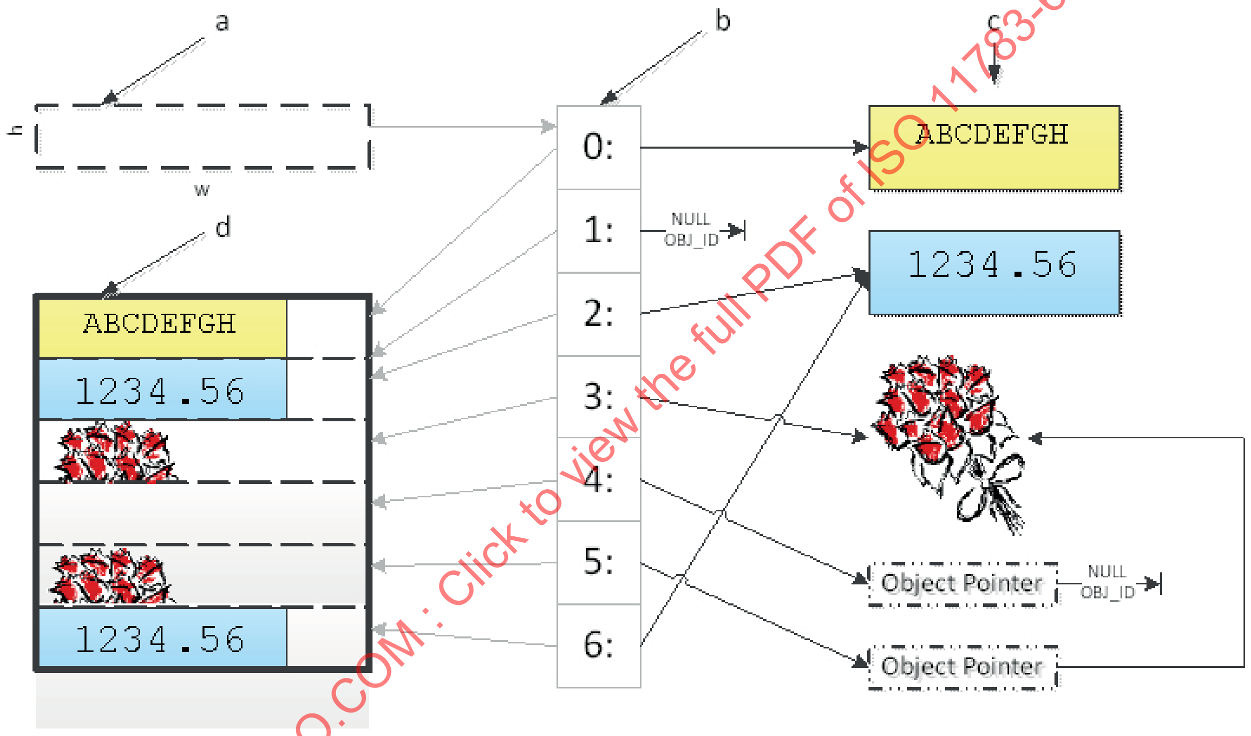
— index value is 255 which means "no item is chosen";

— index value is invalid (greater than the number of items in the list minus 1);

— selected list item is a no-item placeholder (NULL);

— selected list item is an Object Pointer with a value of NULL;

— selected list item is a Container, and the container is in the hidden state.



a Input List object of a specified width and height.
b List item position in the List object.
c Objects referenced by each List item.
d Example of how a list can appear when "Open" for user selection (actual presentation is VT proprietary).

Figure B.4 — Input List object list item example

Allowed commands:

— Enable/Disable Object command;

— Select Input Object command;

— ESC command;

— Change Numeric Value command;

— Change Attribute command;

— Change List Item command;

— Change Size command;

— Get Attribute Value message.

Table B.19 — Input List events

Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Redraw this object.	—
On Enable	Enable/Disable Object command	Mark the input object enabled. If displayed, operator can navigate to it.	Enable/Disable Object Response
On Disable	Enable/Disable Object command	Mark the input object disabled. Even if displayed, the operator cannot select this object for input. VT shall make it clear to the operator that the input field is disabled.	Enable/Disable Object Response
On Input Field Selection	Select Input Object command or operator navigates to input object	The VT shall provide some way for the operator to recognize that the input object is selected (has focus).	VT Select Input Object message
On Input Field De-selection	Select Input Object command or operator navigates off input object or as a result of a disable event	—	VT Select Input Object message
On ESC	Operator aborts input using ESC key or Working Set sends an ESC command	If the input object is not enabled for real time editing, then revert the value of the object to the value before operator began the input. (See Real Time editing in 4.2.) Redraw this object. Refresh parent object.	VT ESC message or ESC response (see 4.6.17).
On Change Value	Change Numeric Value command (to change the list index)	If input object is displayed, redraw object with new value. Refresh parent object.	Change Numeric Value response
On Entry of Value	Operator saving changes by use of the ENTER means regardless of whether or not the value (list index) changed	VT updates the Working Set with new value	VT Change Numeric Value message
On Entry of New Value	Operator saving changes by use of the ENTER means when value (list index) has changed	Working Set is notified by the “On Entry of value” event so no additional notification is required on this event.	—
On Change Attribute	Change Attribute command	If object is visible, refresh.	Change Attribute response.
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size response

Table B.20 — Input List attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=10	3	Object Type = Input List
Width	1	Integer	2	0—65535	4—5	Maximum width of the input field in pixels. Objects or portions of objects outside the defined area are clipped.
Height	2	Integer	2	0—65535	6—7	Maximum height of the input field in pixels. Objects or portions of objects outside the defined area are clipped.
Variable reference	3	Integer	2	0—65534, 65535	8—9	Object ID of a Number Variable object in which to store or retrieve the object's value. If this attribute is set to NULL, the value is stored directly in the value attribute instead.
Value	[4]	Integer	1	0—254, 255	10	Selected list index of this object. Used only if variable reference attribute is NULL. The current list item chosen or 255 to indicate no item is chosen. The first item is at index zero (0).
Number of list items		Integer	1	0—255	11	Number of object references to follow. The size of the list can never exceed this number and this attribute cannot be changed.
Options	[5][b]	Integer	1	0, 1[a] 0—3[b]	12	Logical bits to indicate options. 1 = TRUE. Bit 0 = Enabled. If TRUE the object shall be enabled. If FALSE, the object is disabled. Bit 1 = real time editing. [b] If TRUE the value shall be transmitted to the Working Set Master as it is being changed. (See Real Time editing in 4.2.)
Number of macros to follow		Integer	1	0—255	13	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.

[a] VT Version 3 and prior.

[b] VT Version 4 and later.

Table B.20 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Repeat: {Object ID}		Integer	2	0—65534, 65535	14 + object*2	These objects make up the list. NULL is a no-item placeholder (invisible object) (see A.1.3). The Change List Item command allows objects to be replaced or removed.
Repeat: {Event ID}		Integer	1	0—255	14 + (No. List Items*2)	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3).
{Macro ID}		Integer	1	0—255	15 + (No. List Items*2	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3).

[a] VT Version 3 and prior.

[b] VT Version 4 and later.

B.9 Output field objects

B.9.1 General

There are three types of output field: string, number, and list. They have similar relationships and behaviour, but have different attributes. See Table B.21 to Table B.25.

Allowed Commands (see B.9.4 for Output List object allowed commands):

— Change Background Colour command;

— Change Numeric Value command (excluding Output String object);

— Change String Value command (Output String object only);

— Change Attribute command;

— Change Size command;

— Get Attribute Value message.

Table B.21 — Output field events

Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Redraw this object.	—
On Change Background Colour	Change Background Colour command	If the output field is visible, fill area with background colour and redraw the object with the new background colour.	Change Background Colour Response
On Change Value	Change Numeric Value command or Change String Value command	If output object is displayed, redraw object with new value. Refresh parent object.	Change Numeric Value response or Change String Value response
On Change Attribute	Change Attribute command	If output object is displayed, redraw object with new value. Refresh parent object.	Change Attribute response
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size response

B.9.2 Output String object

This object is used to output a string of text. Displayable characters are given in Table K.2 to Table K.8. Several special formatting characters are permitted in the Output String value and shall be properly interpreted by the VT as specified in 4.6.19.6.

Table B.22 — Output String attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0–65534	1–2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=11	3	Object Type = Output String
Width	1	Integer	2	0–65535	4–5	Maximum width of the output field in pixels. Objects or portions of objects outside the defined area are clipped.
Height	2	Integer	2	0–65535	6–7	Maximum height of the output field in pixels. Objects or portions of objects outside the defined area are clipped.
Background colour	3	Integer	1	0–255	8	Background colour. Used only if the “transparent” bit in the options attribute is cleared. To be applied in the complete rectangle specified by Width and Height.
Font attributes	4	Integer	2	0–65534	9–10	Object ID of a Font Attributes object to use for display formatting of this field.

[a] VT version 3 and prior.

[b] VT version 4 and later.

[c] For VT version 3 and prior, if the Variable reference is not NULL, the Length attribute shall be set to 0.

Table B.22 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Options	5	Bitmask	1	0—3[a] 0—7[b]	11	Logical bits to indicate options. 1 = TRUE. Bit 0 = Transparent. If TRUE, the output field is displayed with background showing through instead of using the background colour attribute. Bit 1 = Auto-Wrap. If TRUE, Auto-Wrapping rules apply (see 4.6.19.5). Bit 2 = Wrap on Hyphen. If TRUE, Auto-Wrapping can occur between a hyphen ($2D_{16}$) and the following character (see 4.6.19.5). Wrap on Hyphen is a modifier to the Auto-Wrap option and is applied only if the Auto-Wrap option is TRUE and ignored if the Auto-Wrap option is FALSE.[b]
Variable reference	6	Integer	2	0—65534, 65535	12—13	Object ID of a String Variable object from which to retrieve the object's value. If this attribute is set to NULL, the string is stored directly in the value attribute instead. If this attribute is not NULL (≤ 65534) the Length and Value attributes are not used.[c]
Justification	7	Integer	1	0—2[a] 0—15[b]	14	Field justification. Indicates how the text string is positioned within the field defined by width and height (see 4.6.19.2). Justification is always done on a graphical (i.e. pixel) basis (see Table B.23). Horizontal Justification Value of Bits 0 — 1 0 = Position Left 1 = Position Middle 2 = Position Right 3 = Reserved Vertical Justification Value of Bits 2 — 3 [b] 0 = Position Top 1 = Position Middle 2 = Position Bottom 3 = Reserved
Length		Integer	2	0—65535	15—16	Maximum fixed length of the output string value in bytes. This can be set to 0 if a variable reference is used.

[a] VT version 3 and prior.

[b] VT version 4 and later.

[c] For VT version 3 and prior, if the Variable reference is not NULL, the Length attribute shall be set to 0.

Table B.22 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Value		String	Length		17-n	Text string to output in the output field. If Length is zero, this attribute is excluded from the record. This attribute shall have the size indicated by the Length attribute — even if a variable reference is used. Pad with spaces as necessary to satisfy length attribute. Can also contain formatting codes as described above. The text string can be 8-bit or WideString (see 4.6.19.7). The Working Set can cause the type to change from an 8-bit String to a WideString or vice versa.
Number of macros to follow		Integer	1	0—255	Depends on size of string	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0—255	Depends on size of string	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3)
{Macro ID}		Integer	1	0—255	Depends on size of string	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)

[a] VT version 3 and prior.

[b] VT version 4 and later.

[c] For VT version 3 and prior, if the Variable reference is not NULL, the Length attribute shall be set to 0.

B.9.3 Output Number object

This object is used to format and output a numeric value based on a supplied integer value. The VT shall use the following equation to format the displayed value:

Displayed value = (value attribute + Offset) × Scaling Factor

Depending on the "Options" attribute presented in Table B.23, displayed values are either truncated or rounded to the number of decimals specified in the "Number of decimals" attribute.

The VT should implement double precision operations to minimize rounding errors.

Table B.23 — Output Number attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=12	3	Object Type = Output Number
Width	1	Integer	2	0—65535	4—5	Maximum width of the output field in pixels. Objects or portions of objects outside the defined area are clipped.
Height	2	Integer	2	0—65535	6—7	Maximum height of the output field in pixels. Objects or portions of objects outside the defined area are clipped.
Background colour	3	Integer	1	0—255	8	Background colour. Used only if the "transparent" bit in the options attribute is cleared. To be applied in the complete rectangle specified by Width and Height.
Font attributes	4	Integer	2	0—65534	9—10	Object ID of a Font Attributes object to use for display formatting of this field.
Options	5	Bitmask	1	0—7[a] 0—15[b]	11	Logical bits to indicate options. 1 = TRUE. Bit 0 = Transparent. If TRUE, the output field is displayed with background showing through instead of using the background colour attribute. Bit 1 = Display leading zeros. If TRUE, fill left to width of field with zeros; justification is applied after filling to the width of the field with zeros. Bit 2 = Display zero value as blank if this bit is TRUE. When this option bit is set, a blank field is displayed if and only if the value of the object is exactly zero. Except when the field is blank, the VT shall always display at least one digit before the decimal point. (examples: 2,2, 0,2) Bit 3 = Truncate. If TRUE the value shall be truncated to the specified number of decimals. Otherwise it shall be rounded off to the specified number of decimals.[b]
Variable reference	6	Integer	2	0—65534, 65535	12—13	Object ID of an integer variable object in which to retrieve the object's raw unscaled value. If this attribute is set to NULL, the value is retrieved directly from the value attribute instead. VT shall scale the value for display.
Value	[12]	Integer	4	0 to 2^32—1	14—17	Raw unsigned value of the output field before scaling (unsigned 32-bit integer). Used only if variable reference attribute is NULL. VT shall scale this value for display.
Offset	7	Signed Integer	4	−2^31 to 2^31—1	18—21	Offset to be applied to the value for display (32-bit signed integer).
Scale	8	Float	4		22—25	Scale to be applied to the value for display.

[a] VT Version 3 and prior.

[b] VT Version 4 and later.

Table B.23 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Number of decimals	9	Integer	1	0—7	26	Specifies number of decimals to display after the decimal point.
Format	10	Boolean	1	0 or 1	27	0 = use fixed format decimal display (####.nn) 1 = use exponential format ([−]###.nnE[+/−]## where n is set by the number of decimals attribute).
Justification	11	Integer	1	0—2[a] 0—15[b]	28	Field justification. Indicates how the number is positioned within the field defined by width and height (see 4.6.19.2). Justification is always done on a graphical (i.e. pixel) basis. Horizontal Justification Value of Bits 0 — 1 0 = Position Left 1 = Position Middle 2 = Position Right 3 = Reserved Vertical Justification Value of Bits 2 — 3[b] 0 = Position Top 1 = Position Middle 2 = Position Bottom 3 = Reserved
Number of macros to follow		Integer	1	0—255	29	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0—255	30...	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3).
{Macro ID}		Integer	1	0—255	31...	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3).

a VT Version 3 and prior.

b VT Version 4 and later.

B.9.4 Output List object

The Output List object, available in VT version 4 and later, is used to show one object out of a set of objects. The object to shown is determined by the Value attribute or a Variable reference.

The VT shall not display anything for the selected item in the following cases:

— index value is 255 which means "no item is chosen";

— index value is invalid (greater than the number of items in the list minus 1);

— selected list item is a no-item placeholder (NULL);

— selected list item is an Object Pointer with a value of NULL;

— selected list item is a Container, and the container is in the hidden state.

Allowed commands:

— Change Numeric Value command;

— Change Attribute command;

— Change List Item command;

— Change Size command;

— Get Attribute Value message.

Table B.24 — Output List events

Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Redraw this object.	—
On Change Value	Change Numeric Value command (to change the list index)	If object is displayed, redraw object with new value. Refresh parent object.	Change Numeric Value response
On Change Attribute	Change Attribute command	If object is visible, refresh.	Change Attribute response
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size response

Table B.25 — Output List attributes and record format

Attribute name	AID	Type	Size Bytes	Range or value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=37	3	Object Type = Output List
Width	1	Integer	2	0—65535	4—5	Maximum width of the output field in pixels. Objects or portions of objects outside the defined area are clipped.
Height	2	Integer	2	0—65535	6—7	Maximum height of the output field in pixels. Objects or portions of objects outside the defined area are clipped.

Table B.25 *(continued)*

Attribute name	AID	Type	Size Bytes	Range or value	Record byte	Description
Variable reference	3	Integer	2	0—65534, 65535	8—9	Object ID of a Number Variable object from which to retrieve the object's value. If this attribute is set to NULL, the value is found directly in the value attribute instead.
Value	[4]	Integer	1	0—254, 255	10	Selected list index of this object. Used only if variable reference attribute is NULL. The current list item chosen or 255 to indicate no item is chosen. The first item is at index zero (0).
Number of list items		Integer	1	0—255	11	Number of object references to follow. The size of the list can never exceed this number and this attribute cannot be changed.
Number of macros to follow		Integer	1	0—255	12	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Object ID}		Integer	2	0—65534, 65535	13 + object*2	These objects make up the list. NULL is a no-item placeholder (invisible object) (see A.1.3 Object relationships). The Change List Item command allows objects to be replaced or removed.
Repeat: {Event ID}		Integer	1	0—255	13 + (No. List Items*2)...	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3).
{Macro ID}		Integer	1	0—255	14 + (No. List Items*2)...	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)

B.10 Output shape objects

B.10.1 General

There are four types of output shape object: line, rectangle, ellipse, and polygon. They have similar relationships and behaviour, but different attributes.

Points contained by these objects shall either be drawn using a square "paintbrush", with the actual point being in the upper left corner of the paintbrush, or the "paintbrush" shall be oriented according to the line direction to get smoother lines. Anti-aliasing is also permitted.

The width of the paintbrush is given by the line width attribute. Lines with width = 0 are not drawn.

The end point is relative to the X-Y start location attributes in the parent object. For line widths greater than 1 pixel, the line thickness can grow to the inside of the object, to the outside of the object, or centred in both directions. The object lines still have to fit into the clipping area of the object without producing any clipped lines.

The end points of the lines shall not be drawn outside of the line length, defined by the surrounding rectangle. For ellipse objects of type segment or section and for polygon objects, adjacent lines shall connect at a single point. See Figure B.5 to Figure B.9, and Table B.26 to Table B.32.

B.10.2 Output Line object

This object outputs a line shape. The starting point for the line is found in the parent object.

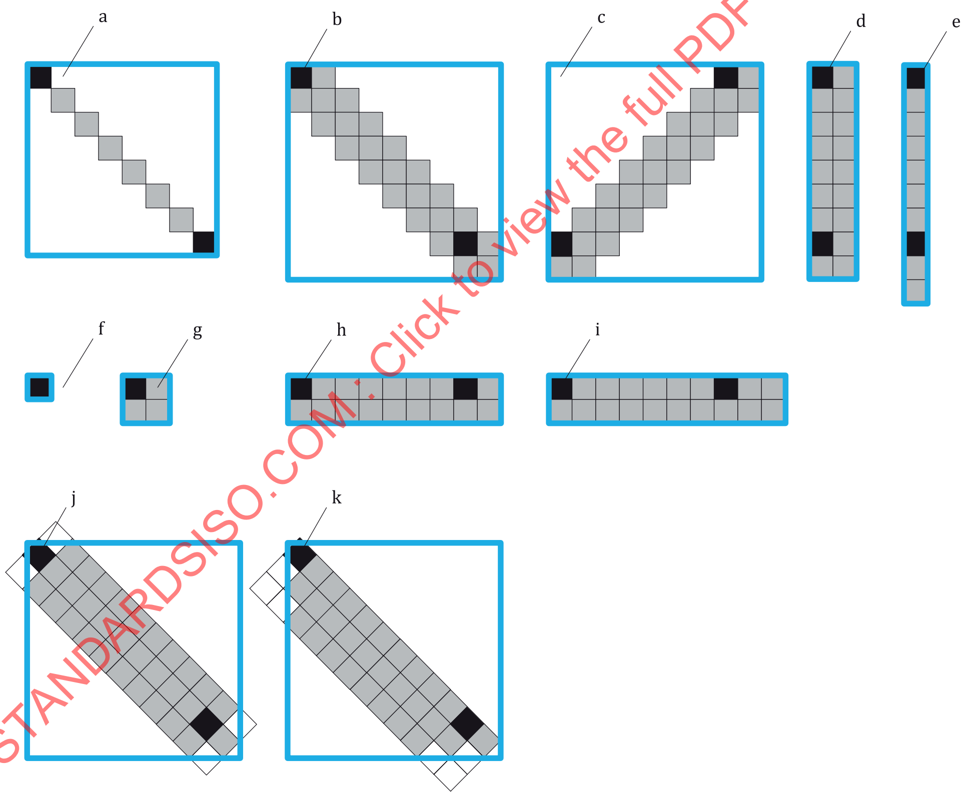
Allowed commands:

— Change End Point command;

— Change Attribute command;

— Change Size command;

— Get Attribute Value message.



Symbol	Width	Height	Direction	Attribute: Line Width
a	8	8	0	1
b	9	9	0	2
c	9	9	1	2
d	2	9	0	2
e	1	10	0	3
f	1	1	Any	≥ 1
g	2	2	Any	≥ 2

h	9	2	0	2
i	10	2	0	3
j	9	9	0	3
k	9	9	0	3

Figure B.5 — Output Line object showing start and end points using different brush sizes

Table B.26 — Output Line events

Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Redraw this object.	—
On Change End Point	Change End Point command	Redraw this object. Refresh parent mask.	Change End Point Response
On Change Attribute	Change Attribute command	Redraw this object. Refresh parent mask.	Change Attribute response
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size response

Table B.27 — Output Line attributes and record format

Attribute Name	Aid	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=13	3	Object Type = Output line
Line attributes	1	Integer	2	0—65534	4—5	Object ID of a Line Attributes object to use for the line attributes.
Width	2	Integer	2	0—65535	6—7	Width in pixels of an enclosing virtual rectangle. NOTE X position plus Width — 1 and Y position plus Height — 1 define the clipping limits.
Height	3	Integer	2	0—65535	8—9	Height in pixels of an enclosing virtual rectangle. NOTE X position plus Width — 1 and Y position plus Height — 1 define the clipping limits.

Table B.27 *(continued)*

Attribute Name	Aid	Type	Size (bytes)	Range or Value	Record byte	Description
Line Direction	4	Integer	1	0 or 1	10	0 = Line is drawn from top left to bottom right of enclosing virtual rectangle StartX = X position of this object StartY = Y position of this object EndX = StartX + Width—Line Width EndY = StartY + Height—Line Width NOTE 1 if EndX < StartX then EndX = StartX NOTE 2 if EndY < StartY then EndY = StartY 1 = Line is drawn from bottom left to top right of enclosing virtual rectangle StartX = X Position of this object StartY = Y Position of this object + Height—Line Width EndX = StartX + Width—Line Width EndY = Y Position of this object NOTE 3 if EndX < StartX then EndX = StartX NOTE 4 if StartY < EndY then StartY = EndY See Figure B.5 for examples of start and end points and line width.

Table B.27 *(continued)*

Attribute Name	Aid	Type	Size (bytes)	Range or Value	Record byte	Description
Number of macros to follow		Integer	1	0—255	11	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0—255	12...	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3).
{Macro ID}		Integer	1	0—255	13...	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3).

B.10.3 Output Rectangle object

This object outputs a rectangle shape. See Figure B.6.

Allowed commands:

— Change Size command;

— Change Attribute command;

— Get Attribute Value message.

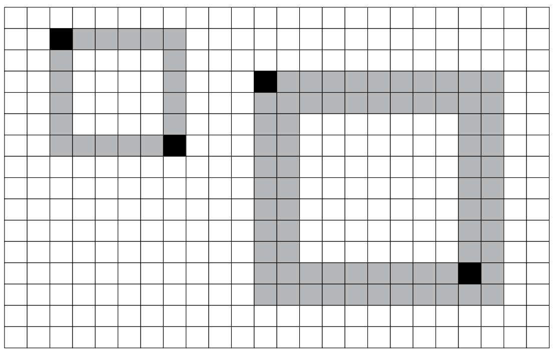


Figure B.6 — Output Rectangle object showing end points using different brush sizes

Table B.28 — Output Rectangle Events

Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Redraw this object.	—
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size response
On Change Attribute	Change Attribute command	Redraw this object, refresh parent mask.	Change Attribute response

Table B.29 — Output Rectangle attributes and record format

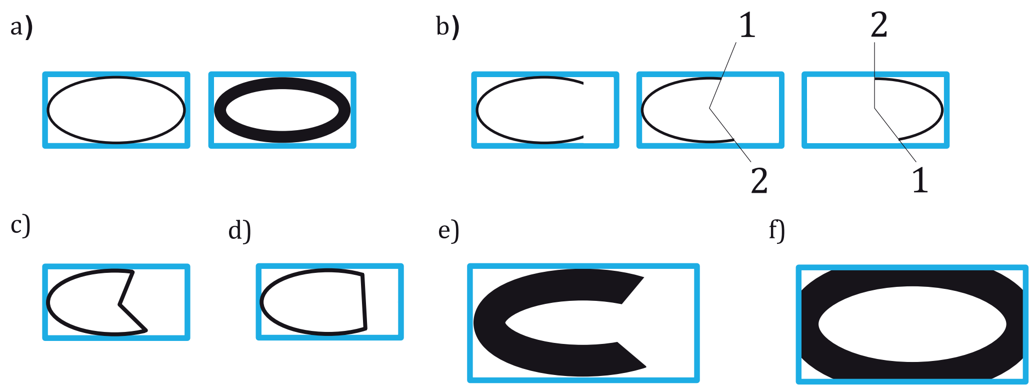
Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=14	3	Object Type = Output Rectangle
Line attributes	1	Integer	2	0—65534	4—5	Object ID of a Line Attributes object to use for the Line Attributes.
Width	2	Integer	2	0—65535	6—7	Width in pixels. (StartX, StartY) to (StartX + Width — 1, StartY + Height – 1) inclusive defines the graphical clipping limits when drawing this object. End point can be calculated as follows: End pointX = StartX + Width – LineWidth End pointY = StartY + Height – LineWidth See Figure B.6 for an example of start and end points and line width.
Height	3	Integer	2	0—65535	8—9	Height in pixels. (StartX, StartY) to (StartX + Width – 1, StartY + Height – 1) inclusive defines the graphical clipping limits when drawing this object. End point can be calculated as follows: End pointX = StartX + Width – LineWidth End pointY = StartY + Height – LineWidth See Figure B.6 for an example of start and end points and line width.

Table B.29 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Line suppression	4	Bitmask	1	0–15	10	Line suppression. These can be combined. 0 = Closed rectangle Bit 0 = 1 = Suppress Top Line (smallest Y value) Bit 1 = 1 = Suppress Right Side Line (largest X value) Bit 2 = 1 = Suppress Bottom Line (largest Y value) Bit 3 = 1 = Suppress Left Side Line (smallest X value) NOTE When drawing a filled rectangle with line suppression, only the pixels that would be on the border of the rectangle are suppressed (not drawn). See Figure 21. Line width shall be taken into account to know the width of the border.
Fill attributes	5	Integer	2	0–65534, 65535	11–12	Object ID of a Fill Attributes object to use for the Fill Attributes or NULL for no fill.
Number of macros to follow		Integer	1	0–255	13	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0–255	14…	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3).
{Macro ID}		Integer	1	0–255	15…	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3).

B.10.4 Output Ellipse object

This object outputs an ellipse or circle shape. Several options are available for modifying the appearance (see Figure B.7).



Key

1	start angle
2	end angle

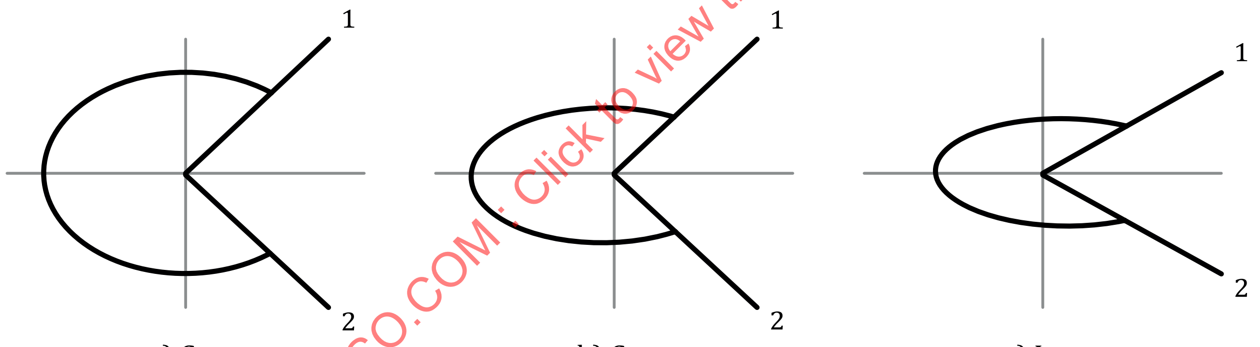
Figure B.7 — Output Ellipse object

Allowed commands:

— Change Size command;

— Change Attribute command;

— Get Attribute Value message.



a) Correct b) Correct c) Incorrect

Key

1	start angle 45°
2	end angle 315°

Figure B.8 — Output Ellipse object — correct and incorrect rendering

Special care should be used when drawing an ellipse which is not a circle. The drawn angle between the start and end angles shall be measured to be accurate. For example, in Figure B.8, the attributes start angle, end angle, and width of all three ellipses are the same. The ellipse on the left is a circle with the height equal to the width, and the ellipses in the centre and on the right both have the height equal to half the width. The angle of the opening should be 90° on all three ellipses. However, the ellipse on the right, was drawn incorrectly using a popular ellipse rendering algorithm that simply scales the full circle to half height. The result is that the angle of the opening is less than 90°.

NOTE Commonly available displays might not have a square aspect ratio for the pixels. The drawing method is not required to compensate for the physical display characteristics.

Table B.30 — Output Ellipse events

Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask refresh for caused by conditions	Redraw this object.	—
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size response
On Change Attribute	Change Attribute command	Redraw this object, refresh parent mask.	Change Attribute response

Table B.31 — Output Ellipse attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=15	3	Object Type = Ellipse
Line attributes	1	Integer	2	0—65534	4—5	Object ID of a Line Attributes object to use for the Line Attributes.
Width	2	Integer	2	0—65535	6—7	Width in pixels of an enclosing virtual rectangle. (StartX, StartY) to (StartX + Width − 1, StartY + Height − 1) *inclusive* defines the graphical clipping limits when drawing this object.
Height	3	Integer	2	0—65535	8—9	Height in pixels of an enclosing virtual rectangle. (StartX, StartY) to (StartX + Width − 1, StartY + Height − 1) *inclusive* defines the graphical clipping limits when drawing this object.
Ellipse type	4	Integer	1	0—3	10	Type of ellipse (see Figure B.7): 0 = Closed Ellipse 1 = Open Ellipse defined by start/end angles 2 = Closed Ellipse Segment 3 = Closed Ellipse Section NOTE 1 If type > 0 and start and end angles are the same, the ellipse is drawn closed. NOTE 2 If type = closed ellipse segment and start and end angle are the same, a single line with width = border width shall be drawn from the centre point to the point on the border defined by the start and end angles.
Start angle	5	Integer	1	0—180	11	Start angle/2 (in degrees) from positive x-axis counter clockwise (90° is straight up). Start and end angles define the arc.
End angle	6	Integer	1	0—180	12	End angle/2 (in degrees) from positive x-axis counter clockwise (90° is straight up). Start and end angles define the arc.

Table B.31 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Fill attributes	7	Integer	2	0—65534, 65535	13—14	Object ID of a Fill attributes object to use for the Fill Attributes or NULL for no fill.
Number of macros to follow		Integer	1	0—255	15	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0—255	16...	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3).
{Macro ID}		Integer	1	0—255	17...	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3).

B.10.5 Output Polygon object

This object outputs a polygon. Four types of polygon are possible: convex, non-convex, complex and open. If the type is not open, the Working Set shall specify the type of polygon as this could affect the efficiency of the fill algorithm. If the type is not known, the type should be set to *complex* since fill algorithms on complex polygons will work on all three fillable types. The VT designer can also choose to implement only the complex fill algorithm and ignore the polygon type attribute. The VT shall use the "even-odd" fill rule. The "non-zero winding" fill rule is not supported.

The start point for the polygon is the first point listed. Polygon is drawn using the points in the list in the order given. At least three points are required for a polygon. The point positions are relative to the upper left-hand corner of the Output Polygon object and the upper left-hand corner of the Output Polygon object is relative to the parent object.

If the polygon type is not "OPEN" and the Working Set does not close the polygon, the VT shall automatically close the polygon by joining the first and last points given.

See Figure B.9.

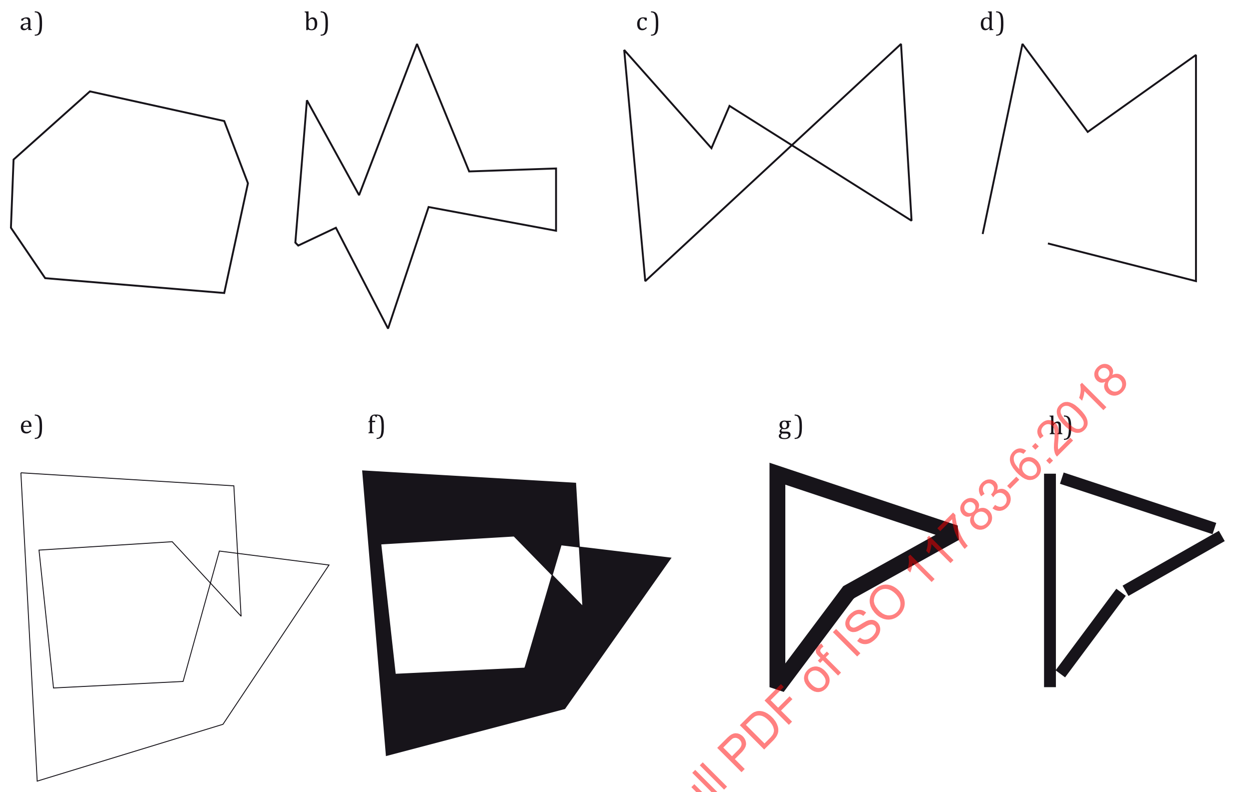


Figure B.9 — Output Polygon types

Allowed commands:

— Change Attribute command;

— Change Size command;

— Change Polygon Point command;

— Change Polygon Scale command;

— Get Attribute Value message.

Table B.32 — Output Polygon events

Event	Caused by	VT behaviour	Message
On Refresh	Change Polygon Point command Change Polygon Scale command Also, see Data Mask Refresh for caused by conditions	Redraw this object.	Change Polygon Point response Change Polygon Scale response
On Change Attribute	Change Attribute command	Redraw this object, refresh parent mask.	Change Attribute response
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size response

Table B.33 — Output Polygon attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=16	3	Object Type = Polygon
Width	1	Integer	2	0—65535	4—5	Width in pixels of an enclosing virtual rectangle. (StartX, StartY) to (StartX + Width − 1, StartY + Height − 1) *inclusive* defines the graphical clipping limits when drawing this object.
Height	2	Integer	2	0—65535	6—7	Height in pixels of an enclosing virtual rectangle. (StartX, StartY) to (StartX + Width − 1, StartY + Height − 1) *inclusive* defines the graphical clipping limits when drawing this object.
Line attributes	3	Integer	2	0—65534	8—9	Object ID of a Line Attributes object to use for the Line Attributes.
Fill attributes	4	Integer	2	0—65534, 65535	10—11	Object ID of a Fill attributes object to use for the Fill Attributes or NULL for no fill.
Polygon type	5	Integer	1	0—3	12	Polygon type. The first three types are useful only if the polygon is to be filled. VT designer can choose to implement only a complex fill algorithm since it will work with all types. Polygon type can only be changed from open to not open or from not open to open. 0 = Convex. On any given horizontal line, only two points on the polygon are encountered. 1 = Non-Convex. On any given horizontal line, more than two points on the polygon edges can be encountered but the polygon edges do not cross. 2 = Complex. Similar to Non-convex but edges cross. Uses Complex Fill Algorithm. 3 = Open. This type cannot be filled.
Number of points		Integer	1	3—255	13	Number of points to follow. Each point is 4 bytes. At least three (3) points shall be listed or this object cannot exist.
Number of macros to follow		Integer	1	0—255	14	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Point X}		Integer	2	0—65535	15+point#*4	X value of a point relative to the top left corner of the polygon.

Table B.33 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
{Point Y}		Integer	2	0—65535	17+point#*4	Y value of a point relative to the top left corner of the polygon.
Repeat: {Event ID}		Integer	1	0—255	15+(num points*4)…	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3).
{Macro ID}		Integer	1	0—255	16+(num points*4)…	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3).

B.11 Output graphic objects

B.11.1 General

There are three types of output graphic object: Output Meter object, Output Linear Bar Graph object and Output Arched Bar Graph object.

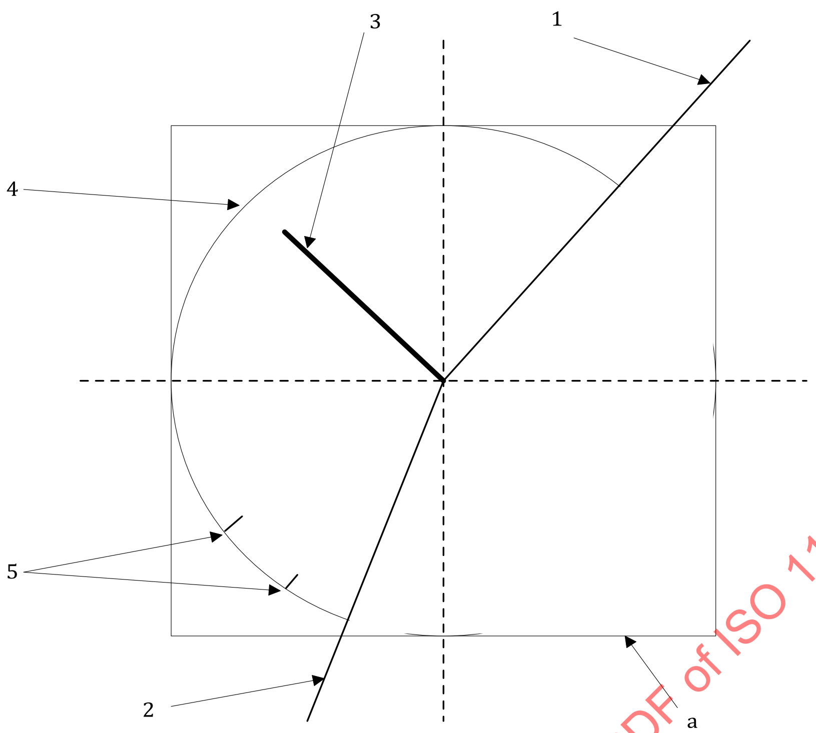
In VT Version 4 and later, if the minimum value is not less than the maximum value, then the object shall be drawn as if the value (or target value) is equal to the minimum value and without regard to the maximum value. Additionally, if the value (or target value) is less than the minimum, the object shall be drawn as if the value (or target value) is equal to the minimum. Likewise, if the value (or target value) is greater than the maximum, the object shall be drawn as if the value (or target value) is equal to the maximum.

In VT Version 3 and prior, the constraints of minimum, value, target, and maximum were not defined.

B.11.2 Output Meter object

This object is a meter. General appearance is left to the VT but the meter is drawn about a circle enclosed within a defined square. The indicated angle attributes are computed from the positive x-axis in a mathematically positive direction (anticlockwise). As with all objects, the VT shall take appropriate action when objects are overlaid so that moving the needle does not corrupt other objects underneath the meter. The position attribute of the meter (in the parent object) always refers to the upper left corner of the enclosing square regardless of orientation. This object is drawn transparent so that objects can be placed underneath to enhance the appearance. See Figure B.10 and Figure B.11 and Table B.34 and Table B.35.

It is recommended that the length of any visible tick marks are 10 % of the width of the meter, with a minimum of 1 pixel (e.g. if the meter is less than 10 pixels in width).



Key

1	start angle
2	end angle
3	needle
4	arc
5	ticks (only 2 ticks are shown for what would be equally spaced from the start angle to the end angle)
a	Meter object cannot exceed boundaries of enclosing square.

Figure B.10 — Output Meter object

Allowed commands:

— Change Numeric Value command;

— Change Attribute command;

— Change Size command;

— Get Attribute Value message.

Table B.34 — Output Meter events

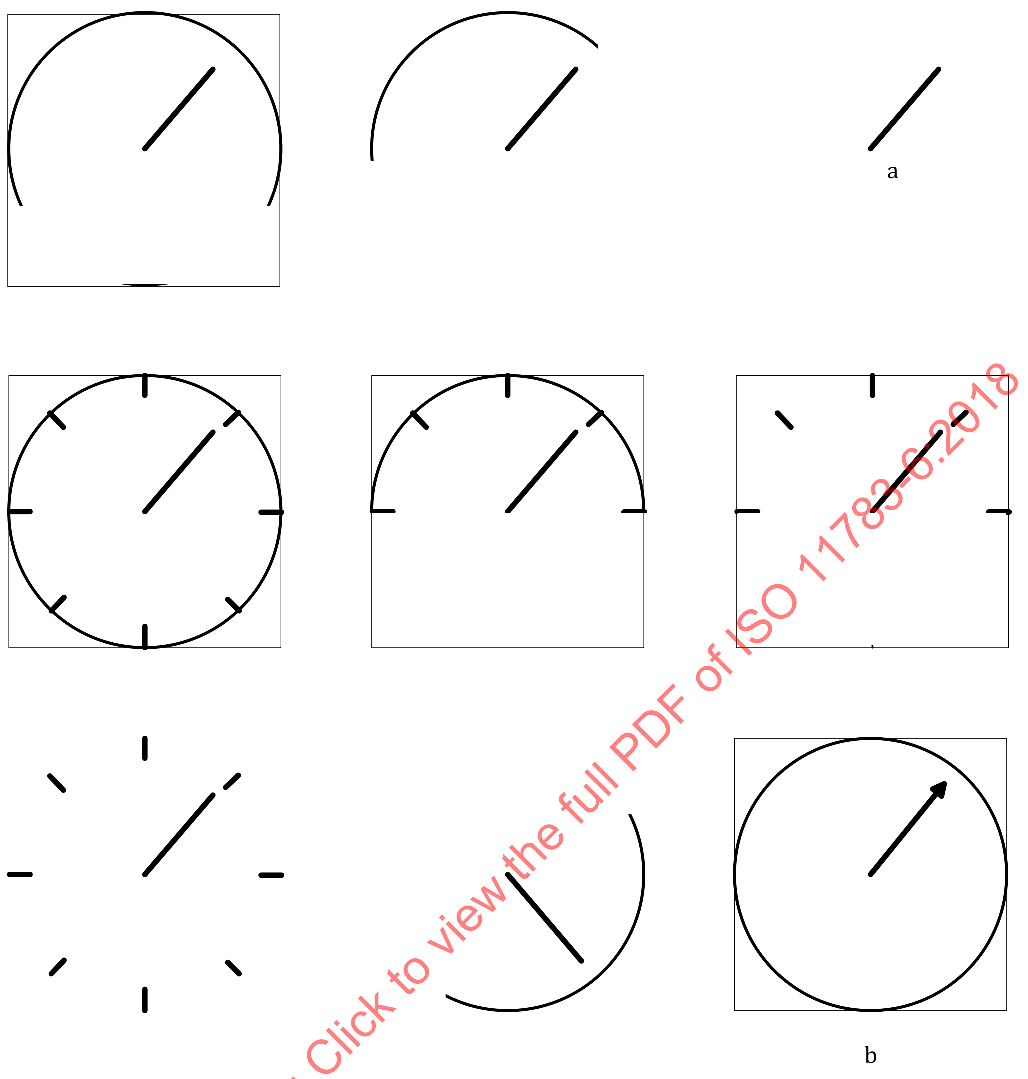
Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Redraw this object.	—
On Change Value	Change Numeric Value command	Redraw this object, refresh parent mask.	Change Numeric Value response
On Change Attribute	Change Attribute command	Redraw this object, refresh parent mask.	Change Attribute response
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size response

Table B.35 — Output Meter attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=17	3	Object Type = Output Meter
Width	1	Integer	2	0—65535	4—5	Maximum width and height of the enclosing square in pixels. Meter object cannot exceed the bounds of this imaginary square.
Needle colour	2	Integer	1	0—255	6	Needle (indicator) colour.
Border colour	3	Integer	1	0—255	7	Border colour (if drawn).
Arc and tick colour	4	Integer	1	0—255	8	meter arc and tick colour (if drawn).
Options	5	Bitmask	1	0—15	9	Logical bits to indicate options. 1 = TRUE. Bit 0 = Draw Arc Bit 1 = Draw Border Bit 2 = Draw Ticks Bit 3 = Deflection Direction. 0 = From minimum to maximum, anticlockwise. 1 = From minimum to maximum, clockwise
Number of ticks	6	Integer	1	0—255	10	Number of ticks to draw about meter arc. If one tick, it is drawn in the middle of the arc. For two or more ticks a tick is placed at each end of the arc and the rest are evenly spaced between them.
Start angle	7	Integer	1	0—180	11	Start angle/2 (in degrees) from positive x-axis anticlockwise (90° is straight up). Start and end angles define the arc. If the start and end angles are the same the meter's arc is closed (360°).
End angle	8	Integer	1	0—180	12	End angle/2 (in degrees) from positive x-axis anticlockwise (90° is straight up). Start and end angles define the arc. If the start and end angles are the same the meter's arc is closed (360°).
Min value	9	Integer	2	0—65535	13—14	Minimum value. Represents value when needle is at start of arc.
Max value	10	Integer	2	0—65535	15—16	Maximum value. Represents value when needle is at end of arc.
Variable reference	11	Integer	2	0—65534, 65535	17—18	Object ID of a Number Variable object in which to retrieve the meter's value. If this attribute is set to NULL, the value is retrieved directly from the value attribute instead. The referenced Number Variable's value shall be in the range 0—65535.
Value	[12]	Integer	2	0—65535	19—20	Current value. Needle position is set by this value. Used only if variable reference is NULL.

Table B.35 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Number of macros to follow		Integer	1	0—255	21	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0—255	22...	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3).
{Macro ID}		Integer	1	0—255	23...	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3).



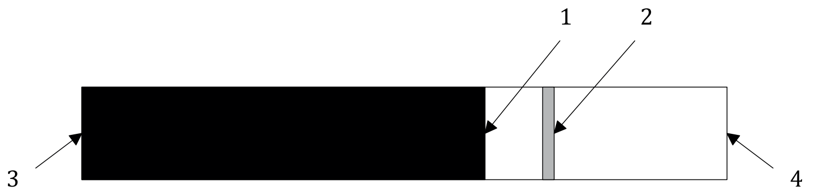
a Needle only.

b Appearance of the Meter object is at the discretion of the VT designer.

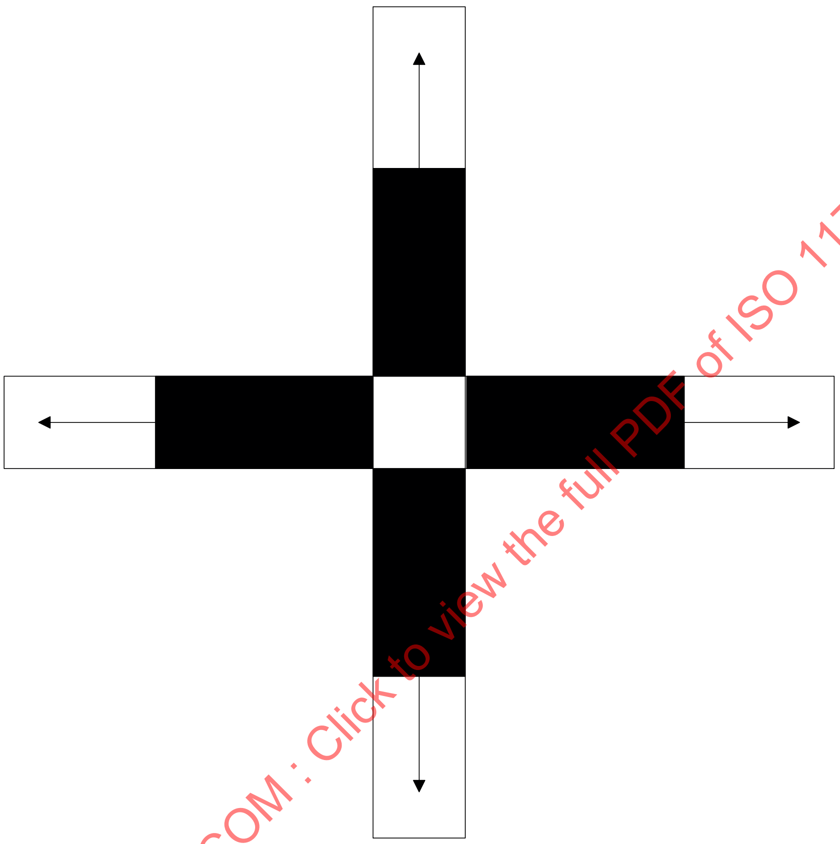
Figure B.11 — Output Meter object — Examples

B.11.3 Output Linear Bar Graph object

This object is a linear bar graph or thermometer. Linear bar graphs are defined by an enclosing rectangle in any one of four orientations. A target value can be optionally marked on the bar graph. The position attribute of the bar graph (in the parent object) always refers to the upper left corner of the enclosing rectangle regardless of orientation.



a) Bargraph cannot exceed the boundary of the enclosing rectangle



b) Bargraph can be in any one of four orientations

Key

1	value
2	target value
3	minimum value
4	maximum value

Figure B.12 — Output Linear Bar Graph — Examples

This object is drawn transparent so that objects can be placed underneath to enhance the appearance. See Figure B.12 and Table B.36 and Table B.37.

Allowed commands:

— Change Numeric Value command;

— Change Attribute command;

— Change Size command;

— Get Attribute Value message.

Table B.36 — Output Linear Bar Graph events

Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Redraw this object.	—
On Change Value	Change Numeric Value command	Redraw this object, refresh parent mask.	Change Numeric Value response
On Change Attribute	Change Attribute command	Redraw this object, refresh parent mask.	Change Attribute response
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size response

Table B.37 — Output Linear Bar Graph attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=18	3	Object Type = Output Output Linear Bar Graph object
Width	1	Integer	2	0—65535	4—5	Maximum width of the enclosing rectangle in pixels. Bar graph cannot exceed the bounds of this imaginary rectangle.
Height	2	Integer	2	0—65535	6—7	Maximum height of the enclosing rectangle in pixels. Bar graph cannot exceed the bounds of this imaginary rectangle.
Colour	3	Integer	1	0—255	8	Bar graph fill and border colour.
Target line colour	4	Integer	1	0—255	9	Target line colour (if drawn).
Options	5	Bitmask	1	0—63	10	Logical bits to indicate which parts to draw: 1 = TRUE Bit 0 = Draw border Bit 1 = Draw target line Bit 2 = Draw ticks Bit 3 = Bar graph type. If this bit is FALSE (0), bar graph is filled. If this bit is TRUE (1), Bar graph is not filled but rather shows the current value as a single line at the proper position within the bar graph. Orientation and direction of the bar graph: Bit 4 = Axis orientation. 0 = vertical (increasing values move parallel to the y-axis with constant X), 1 = horizontal (increasing values move parallel to the x-axis with constant Y) Bit 5 = Direction. 0 = Grows negative (left or down). 1 = Grows positive (right or up).

Table B.37 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Number of ticks	6	Integer	1	0—255	11	Number of ticks to draw along bar graph. If one tick, it is drawn in the middle of the bar graph. For two or more ticks a tick is placed at each end of the bar graph and the rest are evenly spaced between them.
Min value	7	Integer	2	0—65535	12—13	Minimum value.
Max value	8	Integer	2	0—65535	14—15	Maximum value.
Variable reference	9	Integer	2	0—65534, 65535	16—17	Object ID of a Number Variable object in which to retrieve the bar graph's value. If this attribute is set to NULL, the value is retrieved directly from the value attribute instead. The referenced Number Variable's value shall be in the range 0—65535.
Value	[12]	Integer	2	0—65535	18—19	Current value. Used only if variable reference is NULL. Bar graph fills or moves, depending on bar graph type, to a point calculated from this value and min/max values. If value > Max value or value < Min value, the bar graph is filled or shown empty and no error is generated by the VT.
Target value variable reference	10	Integer	2	0—65534, 65535	20—21	Object ID of a Number Variable object in which to retrieve the bar graph's target value. If this attribute is set to NULL, the target value is retrieved directly from the Target value attribute instead. The referenced Number Variable's value shall be in the range 0—65535.
Target value	11	Integer	2	0—65535	22—23	Current target value. Used only if Target value variable Reference attribute is NULL. Target value is displayed as a line on the bar graph to indicate some target or warning level. If Target value > Max value or Target value < Min value, the target line is shown on one of the ends of the bar graph and no error is generated by the VT.

Table B.37 *(continued)*

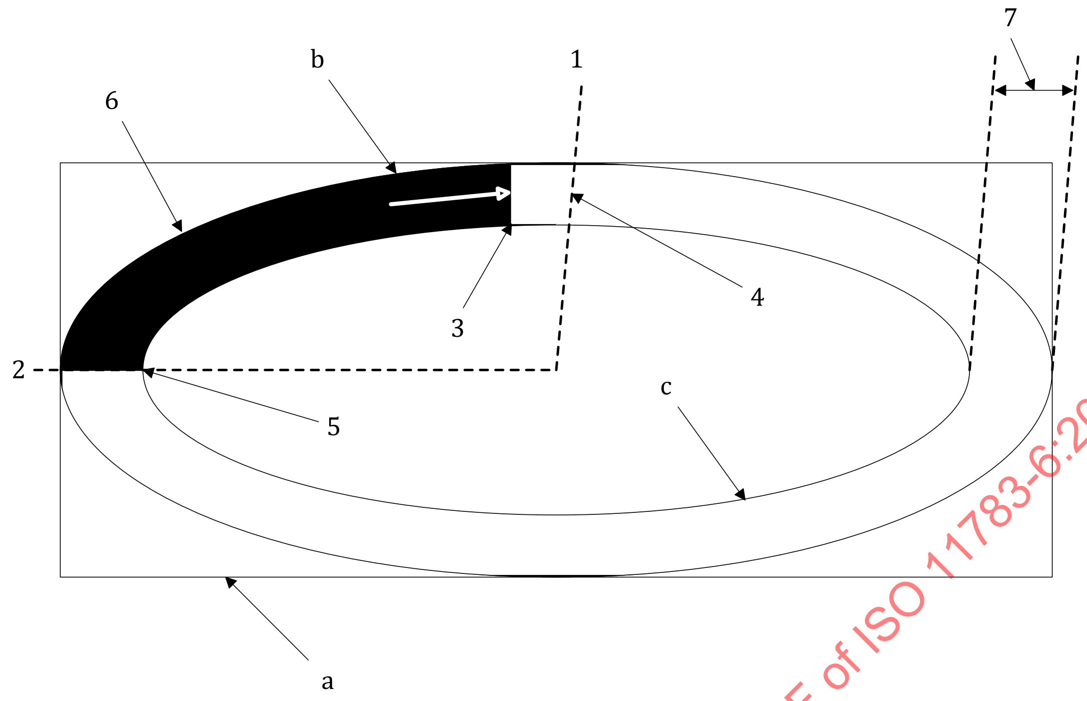
Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Number of macros to follow		Integer	1	0—255	24	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0—255	25…	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3).
{Macro ID}		Integer	1	0—255	26…	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3).

B.11.4 Output Arched Bar Graph object

This object is similar in concept to a linear bar graph but appears arched. Arched bar graphs are drawn about an Output Ellipse object enclosed within a defined rectangle. The indicated angles are computed from the positive x-axis in a mathematically positive direction (anticlockwise). The position attribute of the bar graph (in the parent object) always refers to the upper left-hand corner of the enclosing rectangle regardless of orientation. This object is drawn transparent so that objects can be placed underneath to enhance the appearance.

A Change Size command can cause the “bar graph width” attribute to equal or exceed one half the width or height of the object, however this shall not be cause for pool rejection. The VT can reduce the “bar graph width” value for the purpose of drawing this object. The reduced value is only to supporting drawing the object and shall not be stored in the object.

See Figure B.13 and Table B.38 and Table B.39.



Key

1	start angle	a	Enclosing rectangle. Bar graph cannot exceed boundaries of this rectangle.
2	end angle	b	In this example, bar graph deflection is clockwise.
3	value	c	Visual elements outside the range of start angle to end angle are shown for clarity, but would not be drawn.
4	maximum value		
5	minimum value		
6	border		
7	bar graph width		

Figure B.13 — Output Arched Bar Graph object — Example

Allowed commands:

— Change Numeric Value command;

— Change Attribute command;

— Change Size command;

— Get Attribute Value message.

Table B.38 — Output Arched Bar Graph events

Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Redraw this object.	—
On Change Value	Change Numeric Value command	Redraw this object, refresh parent mask.	Change Numeric Value response
On Change Attribute	Change Attribute command	Redraw this object, refresh parent mask.	Change Attribute response
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size response

Table B.39 — Output Arched Bar Graph attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=19	3	Object Type = Output Arched Bar Graph
Width	1	Integer	2	0—65535	4—5	Maximum width of the enclosing rectangle in pixels. Bar graph cannot exceed the bounds of this imaginary rectangle.
Height	2	Integer	2	0—65535	6—7	Maximum height of the enclosing rectangle in pixels. Bar graph cannot exceed the bounds of this imaginary rectangle.
Colour	3	Integer	1	0—255	8	Bar graph fill and border colour.
Target line colour	4	Integer	1	0—255	9	Target line colour (if drawn).
Options	5	Bitmask	1	0—31	10	Logical bits to indicate which parts to draw. 1 = TRUE. Bit 0 = Draw border. If this bit is TRUE (1), the border lines, including the start and the end line, shall always be drawn. Bit 1 = Draw a target line Bit 2 = Undefined, set to 0 recommended Bit 3 = bar graph type. If this bit is FALSE (0), bar graph is filled. If this bit is TRUE (1), the bar graph is not filled but rather shows the current value as a single line at the proper position within the bar graph. Bit 4 = Deflection of the bar graph around the arc. 0 = anticlockwise and 1 = clockwise
Start angle	6	Integer	1	0—180	11	Start angle/2 (in degrees) from positive x-axis anticlockwise (90° is straight up). Start and end angles define the arc. If the start and end angles are the same, the bar graph's arc is closed (360°).
End angle	7	Integer	1	0—180	12	End angle/2 (in degrees) from positive x-axis anticlockwise (90° is straight up). Start and end angles define the arc. If the start and end angles are the same, the bar graph's arc is closed (360°).
Bar graph width	8	Integer	2	0—65535	13—14	Bar graph width in pixels. Bar graph width should be less than half the total width, or less than half the total height, whichever is least. (See Figure B.13.)
Min value	9	Integer	2	0—65535	15—16	Minimum value.
Max value	10	Integer	2	0—65535	17—18	Maximum value.
Variable reference	11	Integer	2	0—65534, 65535	19—20	Object ID of a Number Variable object in which to retrieve the bar graph's value. If this attribute is set to NULL, the value is retrieved directly from the value attribute instead. The referenced Number Variable's value shall be in the range 0—65535.

Table B.39 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Value	[14]	Integer	2	0—65535	21—22	Current value. Used only if variable Reference attribute is NULL. Bar graph fills or moves, depending on bar graph type, to a point calculated from this value and min/max values. If value > Max value or value < Min value, the bar graph is filled or shown empty and no error is generated by the VT.
Target value variable reference	12	Integer	2	0—65534, 65535	23—24	Object ID of a Number Variable object in which to retrieve the bar graph's target value. If this attribute is set to NULL, the target value is retrieved directly from the Target value attribute instead. The referenced Number Variable's value shall be in the range 0—65535.
Target value	13	Integer	2	0—65535	25—26	Current target value. Used only if target value variable Reference attribute is NULL. Target value is displayed as a line on the bar graph to indicate some target or warning level. If Target value > Max value or Target value < Min value, the target line is shown on one of the ends of the bar graph and no error is generated by the VT.
Number of macros to follow		Integer	1	0—255	27	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0—255	28...	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3).
{Macro ID}		Integer	1	0—255	29...	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3).

B.12 Picture Graphic object

B.12.1 General

This object displays a picture graphic (bitmap). The VT shall scale the picture graphic from the actual width and height to the target width and calculated target height. See Table B.40 and Table B.41.

Allowed commands:

— Change Attribute command;

— Get Attribute Value message.

Table B.40 — Picture Graphic events

Event	Caused by	VT behaviour	Message
On Refresh	A change in the Opaque/Transparent or the Flashing Option bits Also see Data Mask Refresh for caused by conditions	Redraw this object.	—
On Change Attribute	Change Attribute command	Redraw this object, refresh parent mask.	Change Attribute response

Table B.41 — Picture Graphic attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=20	3	Object Type = Picture Graphic
Width	1	Integer	2	0—65535	4—5	Target width in pixels of the picture graphic. The height of the picture graphic is calculated from the Actual width/height and this attribute to keep the same aspect and avoid distortion.
Actual width	[4]	Integer	2	0—65535	6—7	Actual width in pixels of the picture graphic raw data. VT shall scale the graphic to the size given by the width attribute.
Actual height	[5]	Integer	2	0—65535	8—9	Actual height in pixels of the picture graphic raw data. VT shall scale the graphic to the size given by the width attribute.
Format	[6]	Integer	1	0—2	10	Picture graphic type: 0 = Monochrome; 8 pixels per byte. Each bit represents a colour palette index of 0 or 1. ("White" colour can vary with display hardware) 1 = 4 bit colour; 2 colour pixels per byte. Each nibble (4 bits) represents a colour palette index of 0 through 15. 2 = 8 bit colour; 1 colour pixel per byte. Each byte represents a colour palette index of 0 through 255. See Table A.4.

Table B.41 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Options	2	Bitmask	1	0—7	11	Bit 0: 0 = Opaque, 1 = Transparent. If opaque, all pixels are drawn in indicated colour. Background objects do not show through. If transparent, pixels in the bitmap that have the transparency colour should show the colour of the background or objects underneath this picture graphic instead. Bit 1: 0 = Normal, 1 = Flashing. Flash style and rate determined by VT design. Bit 2: 0 = Raw data, 1 = Run-Length Encoded data (see B.12.2). This bit cannot be changed during runtime by Change Attribute command. Any change in this bit shall be ignored by the VT.
Transparency colour	3	Integer	1	0—255	12	Pixels in the bitmap that have this colour index are transparent (background shows through).
Number of bytes in raw data		Integer	4	0 to 2^32—1	13—16	Number of bytes in the raw data.
Number of macros to follow		Integer	1	0—255	17	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.

Table B.41 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Repeat: {raw data}		Integer	1	0—255	18...	Raw bytes of graphic data. Bytes shall be interpreted according to the format and options attributes. For an explanation of raw data format (see B.12.2). If monochrome bitmap, each byte contains the colour indices for 8 pixels beginning at the left with the most significant bit. If 4-bit colour bitmap, each byte contains the colour indices for two pixels beginning at the left with the most significant nibble. If 8-bit colour bitmap, each byte contains the colour index for one pixel. Bitmap data are always interpreted left to right, top to bottom of the display. Unused bits at the end of a line are ignored.
Repeat: {Event ID}		Integer	1	0—255	Depends on size of bitmap data	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3).
{Macro ID}		Integer	1	0—255	Depends on size of bitmap data	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3).

B.12.2 Picture Graphic object raw data format and compression

The raw data attribute of the Picture Graphic object contains pixel information line by line, left to right and downwards. If the width of the object is such that the data does not end evenly at the end of a byte, the unused portion of the byte at the end of each line is filled with pixel values equal to zero (0). The VT ignores unused parts of a byte at the end of a line.

EXAMPLE If the size of the object is 10 pixels wide by two lines high, the format is monochrome and each line is filled with white coloured pixels. The unused 6 bits in the second byte of each line would be filled with zero and the raw data would be FF_{16},$C0_{16}$,FF_{16},$C0_{16}$. Similar logic is applied for four-bit colour graphics where, if the width is odd, the least significant nibble of the last byte on each line is set to zero.

If the data are longer than expected after all of the rows and columns of pixels have been defined, then the VT shall ignore all extra data bytes. However, if the data are shorter than expected leaving some pixels undefined, then the VT shall report an error to the Working Set. This error shall be reported with either the End of Object Pool response (see C.2.5) or the VT Change Active Mask message (see H.14).

In order to reduce the amount of data transmitted by the Working Set and maintained by the VT it is recommended that the smallest version of a picture graphic be transmitted. Therefore, a run-length encoding scheme can be used to compress the picture graphic data. Care should be taken by Working Set designers as this algorithm can actually increase the size of the picture graphic data when the object is complex. In this case, it is recommended that raw data be transmitted instead and Bit 2 of the options attribute should be cleared.

The compression algorithm is simple and works as follows. Data are transmitted in two-byte pairs with the first byte representing the number of times the value byte repeats and the second byte representing the value to repeat. For example, for a raw data sequence of 0,0,0,0,0,0,3,3,3,1,1,2, the compressed data would be transmitted as 6,0,3,3,2,1,1,2. This example gives a compression of 33 %. If the first byte in a pair has the value zero, the second byte is ignored.

The run-length algorithm is chosen because picture graphic data can be easily compressed and uncompressed in real time without the need for a buffer in the VT.

B.13 Variable objects

B.13.1 General

Variables are used to store a value that can be referenced and used by other objects. There are two types of variable object: number and string. Variables are referenced only, never directly included as a child in a parent object. See Table B.42.

Table B.42 — Variable events

Event	Caused by	VT behaviour	Message
On Change Value	Change Numeric Value command or Change String Value command	Redraw all objects that are currently displayed and reference this object. Refresh parent object.	Change Numeric Value response or Change String Value response

B.13.2 Number Variable object

A number variable holds a 32-bit unsigned integer value. See Table B.43.

Allowed commands:

— Change Numeric Value command (Number Variable object only);

— Get Attribute Value message.

Table B.43 — Number Variable attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=21	3	Object Type = Number Variable
Value	[1]	Integer	4	0 to 2^32—1	4—7	32-bit unsigned integer value.

B.13.3 String Variable object

A String Variable holds a fixed length string. Strings shorter than the length attribute should be padded with space characters. The maximum length attribute cannot be changed once the variable has been defined. See Table B.44.

Allowed commands:

— Change String Value command (String Variable object only);

— Get Attribute Value message.

Table B.44 — String Variable attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=22	3	Object Type = String Variable
Length		Integer	2	0—65535	4—5	Maximum fixed length of the string value in bytes.
Value		String			6...	String of characters. Pad with spaces as necessary to satisfy length attribute. The text string can be 8-bit or WideString (see 4.6.19.7). The Working Set can cause the type to change from an 8-bit String to a Wide-String or vice versa.

B.14 Attribute objects

B.14.1 General

Attribute objects are used to hold common attributes for other objects. Attribute objects are referenced only, never directly included as a child in a parent object. There are four types of attribute object: font, line, fill and input. See Table B.45 and Table B.46.

B.14.2 Font Attributes object

This object holds attributes related to fonts.

The Working Set designer can change the Font Attributes using either the Change Font Attributes command, the Change Attribute command, or by transferring a new object. The designer should be aware that some uses of the Change Attribute command can cause the object pool to become invalid (e.g. if the Font size and Font style attributes are changed to a combination that the VT does not support). The Change Font Attributes command can be used to achieve the desired results without the invalid pool condition.

Allowed commands:

— Change Font Attributes command;

— Change Attribute command;

— Get Attribute Value message.

Table B.45 — Font Attributes events

Event	Caused by	VT behaviour	Message
On Change Font Attributes	Change Font Attributes command	Redraw all objects that are currently displayed and reference this object. Refresh parent object.	Change Font Attributes response
On Change Attribute	Change Attribute command	Redraw all objects that are currently displayed and reference this object. Refresh parent object.	Change Attribute response

Table B.46 — Font Attributes attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=23	3	Object Type = Font Attributes
Font colour	1	Integer	1	0—255	4	Text colour.
Font size	2	Integer	1	0 — 14[a] 0—14 or 8—N[b]	5	Font size If Non-Proportional Font (refer to Font style bit 7) Font size value = pixel width × pixel height 0 = 6 × 8 1 = 8 × 8 2 = 8 × 12 3 = 12 × 16 4 = 16 × 16 5 = 16 × 24 6 = 24 × 32 7 = 32 × 32 8 = 32 × 48 9 = 48 × 64 10 = 64 × 64 11 = 64 × 96 12 = 96 × 128 13 = 128 × 128 14 = 128 × 192 If Proportional font (refer to font style bit 7): 8 to N This attribute represents the height of the font in pixels in the range 8 up to and including the value N, where N is the largest supported font height as identified in the Font size value of Get Text Font Data response. Width of each character is variable.[b]
Font type	3	Integer	1	0, 1, 255[a] 0—2, 4, 5, 7, 240—255[b]	6	See Table K.1. If the Font Attributes object applies to a WideString the Font type is ignored (see 4.6.19.7).

[a] VT Version 3 and Prior.

[b] For version 4 and later VTs.

[c] Inverting is to exchange background and pen colours. The rules for background transparency shall be applied.

Table B.46 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Font style	4	Bitmask	1	0—127[a] 0—255 [b]	7	Font style. These can be combined. 0 = Normal Text (default) Bit 0 = 1 = Bold Bit 1 = 1 = Crossed Out Bit 2 = 1 = Underlined Bit 3 = 1 = Italic Bit 4 = 1 = Inverted[c] Bit 5 = 1 = Flashing between Inverted and styles set by bits 0—3 Bit 6 = 1 = Flash both the background and the foreground between Hidden and styles set by bits 0—4. Bit 6 has priority over bit 5. In other words, the entire text object is hidden on the "hidden" cycle. Bit 7 = 1 = Proportional font rendering (if this bit is zero, use non-proportional font rendering).[b]
Number of macros to follow		Integer	1	0—255	8	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0—255	9…	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3).
{Macro ID}		Integer	1	0—255	10…	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3).

[a] VT Version 3 and Prior.

[b] For version 4 and later VTs.

[c] Inverting is to exchange background and pen colours. The rules for background transparency shall be applied.

B.14.3 Line Attributes object

This object holds Line Attributes related to output shape objects. See Table B.47 and Table B.48 and record format and Figure B.15.

NOTE The end point of a line can be calculated, but is not necessarily drawn when the Line Attribute is applied.

Allowed commands:

— Change Line Attributes command;

— Change Attribute command;

— Get Attribute Value message.

Table B.47 — Line Attributes events

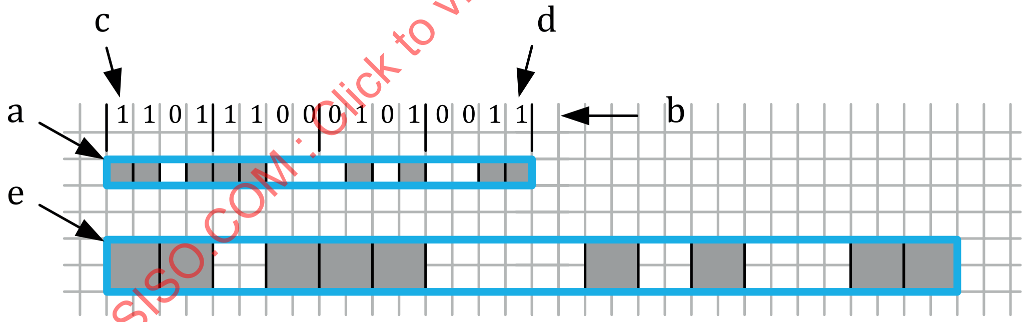
Event	Caused by	VT behaviour	Message
On Change Line Attributes	Change Line Attributes command	Redraw all objects that are currently displayed and reference this object. Refresh parent object.	Change Line Attributes response
On Change Attribute	Change Attribute command	Redraw all objects that are currently displayed and reference this object. Refresh parent object.	Change Attribute response

Table B.48 — Line Attributes attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0–65534	1–2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=24	3	Object Type = Line Attributes
Line colour	1	Integer	1	0–255	4	Pen colour.
Line width	2	Integer	1	0–255	5	Pen thickness in pixels. Lines are drawn with a square paintbrush of this size. See Figure B.14.
Line art	3	Bitmask	2	0–65535	6–7	Bit pattern art for line. Each bit represents a paintbrush spot. Zero (0) bits are skipped (background colour) and one (1) bits are drawn in the line colour. Each bit is the size of the current paintbrush. For example, 00110011 would represent two skipped paintbrush spots followed by two paintbrush spots drawn and so on. See Figure B.15.

Table B.48 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Number of macros to follow		Integer	1	0—255	8	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0—255	9...	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3)
{Macro ID}		Integer	1	0—255	10...	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)



a Output Line object (width = 16, height = 1, line attribute: line width = 1, line attribute: line art = $DC53_{16}$).

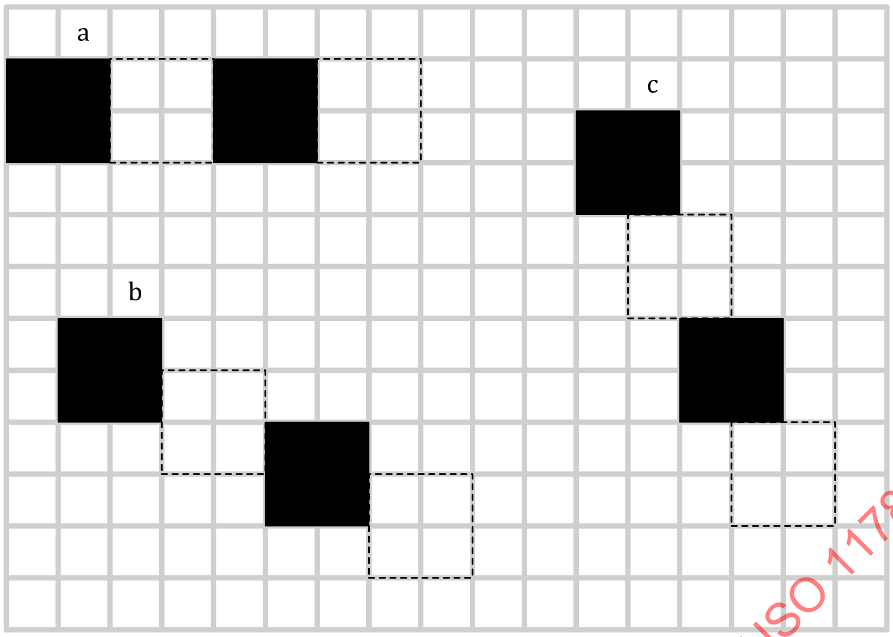
b Line Attribute binary value superimposed above the rendered Output Line object to show relationship.

c Most significant bit of line attribute.

d Least significant bit of line attribute.

e Output Line object (width = 32, height = 2, line attribute: line width = 2, line attribute: line art = $DC53_{16}$).

Figure B.14 — Effect of Line Attribute — Example of same line art with different width



a Output Line object (width = 8, height = 2, line attribute: line width = 2, line attribute: line art = $AAAA_{16}$).
b Output Line object (width = 8, height = 5, line attribute: line width = 2, line attribute: line art = $AAAA_{16}$).
c Output Line object (width = 5, height = 8, line attribute: line width = 2, line attribute: line art = $AAAA_{16}$).

Figure B.15 — Effect of Line Attribute — Example pattern: 1010…

B.14.4 Fill Attributes object

This object holds attributes related to filling output shape objects. See Table B.49 and Table B.50.

Allowed commands:

— Change Fill Attributes command;

— Change Attribute command;

— Get Attribute Value message.

Table B.49 — Fill Attributes events

Event	Caused by	VT behaviour	Message
On Change Fill Attributes	Change Fill Attributes command	Redraw all objects that are currently displayed and reference this object. Refresh parent object.	Change Fill Attributes response
On Change Attribute	Change Attribute command	Redraw all objects that are currently displayed and reference this object. Refresh parent object.	Change Attribute response

Table B.50 — Fill Attributes attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=25	3	Object Type = Fill Attributes
Fill type	1	Integer	1	0—3	4	0 = no fill 1 = fill with line colour 2 = fill with specified colour in fill colour attribute 3 = fill with pattern given by fill pattern attribute
Fill colour	2	Integer	1	0—255	5	Colour for fill if Fill type = 2. Ignored for all other Fill type values.
Fill pattern	3	Integer	2	0—65534, 65535	6—7	Object id of a Picture Graphic object to use as a Fill pattern. Ignored if Fill type <> 3. To change from Fill Type 0, 1 or 2 to Fill Type 3, the Working Set shall modify the Fill Pattern attribute first and the Fill Type attribute second to avoid errors in the VT. If this order is not followed, the behaviour of the VT cannot be predicted and is proprietary. If the Fill Type is 3 and the Fill Pattern attribute is the NULL, no fill shall be performed by the VT. IMPORTANT In order to simplify monochrome and 16-colour VTs, the referred Picture Graphic object shall have a pattern buffer which does not contain any unused bits. Therefore, the width of the Picture Graphic object with Format = 0 (monochrome) shall be evenly divisible by 8. The Picture Graphic object with Format = 1 (16-colour) shall be evenly divisible by 2 (see Example in B.12.2). If these conditions are not met the VT shall report an error to the Working Set. This error shall be reported with either the End of Object Pool response (see C.2.5) or the VT Change Active Mask (see H.14).

Table B.50 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Number of macros to follow		Integer	1	0–255	8	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0–255	9...	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3).
{Macro ID}		Integer	1	0–255	10...	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3).

B.14.5 Input Attributes object

This object defines the valid or invalid characters for an Input String object. The VT shall check this object for valid characters and not permit operator entry of invalid characters into the input field. This object is referenced by an Input String object. See Table B.51 and Table B.52.

If the Input String object which references this object does not contain an 8-bit string, or the Input String object references a String Variable that does not contain an 8-bit string, then no validation shall be performed.

Allowed commands:

— Change String Value command;

— Get Attribute Value message.

Table B.51 — Input Attributes events

Event	Caused by	VT behaviour	Message
On Change Value	Change String Value command		Change String Value response

Table B.52 — Input Attributes attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=26	3	Object Type = Input Attributes
Validation type	[1]	Integer	1	0—1	4	0 = valid characters are listed 1 = invalid characters are listed
Length		Integer	1	0—255	5	Length of validation string in bytes The Validation String shall be an 8-bit String.
Validation string		String			6...	String containing all valid or invalid character codes (depends on validation type attribute).
Number of macros to follow		Integer	1	0—255	Depends on size of string	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0—255	Depends on size of string	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3).
{Macro ID}		Integer	1	0—255	Depends on size of string	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3).

B.14.6 Extended Input Attributes object

The Extended Input Attributes object, available in VT version 4 and later, defines the valid or invalid characters for an Input String object. The VT shall check this object for valid characters and not permit operator entry of invalid characters into the input field. This object is referenced by an Input String object. See Table B.53.

If the Input String object which references this object does not contain a WideString, or the Input String object references a String Variable object that does not contain a WideString, then no validation shall be performed.

The character ranges defined in this object can include characters which are not supported by the VT. The VT shall ignore the unsupported characters but still validate against the remaining characters.

Allowed command:

— Get Attribute Value message.

Table B.53 — Extended Input Attributes attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=38	3	Object Type = Extended Input Attributes
Validation type	[1]	Integer	1	0—1	4	0 = valid characters are listed 1 = invalid characters are listed
Number of code planes to follow		Integer	1	1—17	5	Number of code planes with valid/invalid characters. For each code plane the plane number and an array of character ranges are listed.
Repeat: {Code plane number}		Integer	1	0—16	6...	Code plane to which the character ranges belong. 0: characters $00000_{16}..0FFFF_{16}$ 1: characters $10000_{16}..1FFFF_{16}$ 2: characters $20000_{16}..2FFFF_{16}$ etc.
{Number of character ranges to follow}		integer	1	1—255	7..	Number of character ranges. Each character range consists of two WideChars (first character and last character where the first character ≤ last character). Depending on validation type (byte 4) the ranges indicate either valid or invalid characters.
Repeat: {{First character}}		integer	2	0—65535	8..	First character in the range.
{{Last character}}		integer	2	0—65535	10..	Last character in the range.

EXAMPLE An Extended Input Attribute object (object id 1234_{16}) limiting the input characters to the following ranges:

00041_{16}-$0004F_{16}$	; Code plane 0
00061_{16}-$0006F_{16}$	; Code plane 0
$1AB12_{16}$-$1AB1F_{16}$	; Code plane 1

The object shall be encoded as follows:

34_{16}, 12_{16},	; object id
26_{16},	; Type
00_{16},	; validation type
02_{16},	; Two code planes
00_{16},	; Code plane 0
02_{16},	; Two character ranges
41_{16}, 00_{16}, $4F_{16}$, 00_{16},	; Range 1
61_{16}, 00_{16}, $6F_{16}$, 00_{16},	; Range 2
01_{16},	; Code plane 1
01_{16},	; One character range
12_{16}, AB_{16}, $1F_{16}$, AB_{16},	; Range 1

B.15 Object Pointer object

Refer to Clause 4.6.11.5 Object pointer for information on this object. See Table B.54 and Table B.55.

Allowed commands:

— Change Numeric Value command;

— Get Attribute Value message.

Table B.54 — Object Pointer events

Event	Caused by	VT behaviour	Message
On Change Value	Change Numeric Value command	Hide the prior object and show the new one. Refresh the parent object	Change Numeric Value response

Table B.55 — Object Pointer attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=27	3	Object Type = Object Pointer object
Value	[1]	Integer	2	0—65534, 65535	4—5	Object ID of a referenced object or the NULL Object ID.

B.16 Macro object

Macros are used to define a list of commands that can be referenced by an event or executed using the Execute Macro command (on version 4 or later VTs) or the Execute Extended Macro command (on version 5 and later VTs). A Macro is defined by a series of one or more command packets. (See 4.6.11.4).

It is the Working Set responsibility to ensure that the macros, prior to execution, are consistent with the object pool (e.g. cannot reference missing objects).

NOTE Command packets are defined in Annex F but not all commands are allowed in a Macro.

See Table B.56.

Allowed commands:

— Execute Macro command;

— Execute Extended Macro command;

— Get Attribute Value message.

Table B.56 — Macro attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—255[a] 0—65534[b]	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=28	3	Object Type = Macro
Number of bytes to follow		Integer	2	0—65535	4—5	Number of bytes to follow. For each command, if the command packet is less than 8 bytes (e.g. Change String Value command on a two byte string), the remaining bytes shall be set to FF_{16} to pad the packet to an 8 byte boundary.
Repeat: {Command}					6—n	Command message packets with each packet making up a command. Only commands listed in Annex F are allowed. Use formats from Annex F.

[a] VT version 4 and prior.

[b] VT version 5 and later.

B.17 Colour Map object

The Colour Map object, optionally available in VT version 4 and 5, and mandatory in VT version 6 and later, allows the Working Set designer to alter the transformation of the VT colour index values to the defined RGB value. This provides a mechanism where the colours table can be changed at run-time. A Working Set, in using a few colour objects for various backgrounds and borders can then easily alter the presentation.

The object pool can contain more than one Colour Map object. After the pool is loaded the VT uses the default Colour Map. Upon receipt of a valid Select Colour Map or Palette command the VT changes colour map accesses of the active colour palette for this Working Set. Every object of a Working Set's pool shall be displayed with that Working Set's Colour Map. VT version 6 and later supports changing from the VT standard colour palette to a Working Set defined colour palette using the Colour Palette object (see 4.8 and B.26).

The Colour Map object contains the definitions for each of the valid colour index values. Upon selection of a Colour Map, the resulting Colour Map values shall be valid for the VTs capabilities or the VT will indicate the failure in the Select Colour Map or Palette response message. Therefore, a monochrome VT maintains two entries in the Colour Map object, where a 256 colour VT maintains 256 entries in the Colour Map object. The Colour Map object can have capabilities beyond the VT (e.g. a 256 entry Colour Map object can be uploaded for a 2-colour VT, which will only access indices 0 and 1), but the Colour Map object shall not have fewer colour indices than the VT capabilities.

A typical VT implementation defines the default palette as shown in Table A.4. The subscript into this array of values is the colour index. The Colour Map object provides one level of indirection to the RGB table. Figure 32 shows a sample presentation before and after colours 0 and 1 have been reversed by selecting a Colour Map object with the values [1, 0, ...].

Allowed command:

— Get Attribute Value message.

Table B.57 — Colour Map attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=39	3	Object Type = Colour Map
Number of colour indexes to follow		Integer	2	2, 16, 256	4—5	Indicates the number of Colour Map entries to follow. Allowable values: 2 for a VT with graphic type 0 (monochrome) 16 for a VT with graphic type 1 (16 colour) 256 for a VT with graphic type 2 (256 colour)
Colour Map		Integer	variable	0—255	6—n	VT Colour to be shown for VT colour index values VT graphic type 0: length = 2 for colour index 0, 1 VT graphic type 1: length = 16 for colour index 0 — 15 VT graphic type 2: length = 256 for colour index 0—255

B.18 Graphics Context object

The Graphics Context Object, optionally available in VT version 4 and later, and mandatory in VT version 6 and later, is a bitmap with a canvas and a viewport that can be manipulated by the Working Set at run time. If it is not supported by the VT but is referenced in the object pool, it is handled like an Object Pointer pointing to the NULL object ID (e.g. it is not drawn and occupies a selectable place in an opened Input List). No additional memory is allocated in the VT in this case and the storage space of the GCO(s) can be omitted in the (see D.2).

The canvas of the object (the drawing area) can be changed and is remembered by the object even when it is not physically on the screen. In other words, this object has a memory and the pixels of the canvas persist even when the object is removed from the display or another mask is selected. This allows a Working Set to do run-time drawing to the VT screen. This would be useful for precision farming applications, where, for example, the designer could draw a swathe behind the moving implement image.

The memory required for the object's bitmapped contents can be directly calculated from the canvas size. Changes to the canvas size are not permitted unless an entirely new object is uploaded. The "viewport" defines the portion of the canvas that is visible and thus the display size of the VT object (i.e. on a Data Mask). By anchoring the viewport as a child object in the parent mask or container, it is possible to easily and efficiently pan the underlying object contents within the viewport. The size of the viewport can be modified at run-time.

The contents of this object are never stored by the Store Version command. Working Sets can copy the canvas to a Picture Graphic before storing a version of the object pool if it is desired to store what was drawn.

A single Graphics Context command with several sub-commands is defined to allow simple, consistent and efficient modification of the contents of the graphics Context. The Graphics Context commands can also be contained within a Macro to permit even more efficient updates. Most commands can be carried by one CAN packet.

The current drawing Context, or attributes of the Graphics Context object are always remembered. Therefore it is only necessary to set an attribute once when it is intended to be used more than once (refer to example below). A graphics cursor is defined to indicate the next X/Y location at which to start drawing. Graphics commands can move the graphics cursor to a new location. Therefore drawing can be thought of as a set of procedural commands as shown in Figure B.17.

Instructions	Graph Result

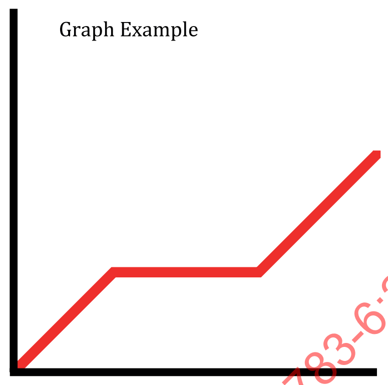
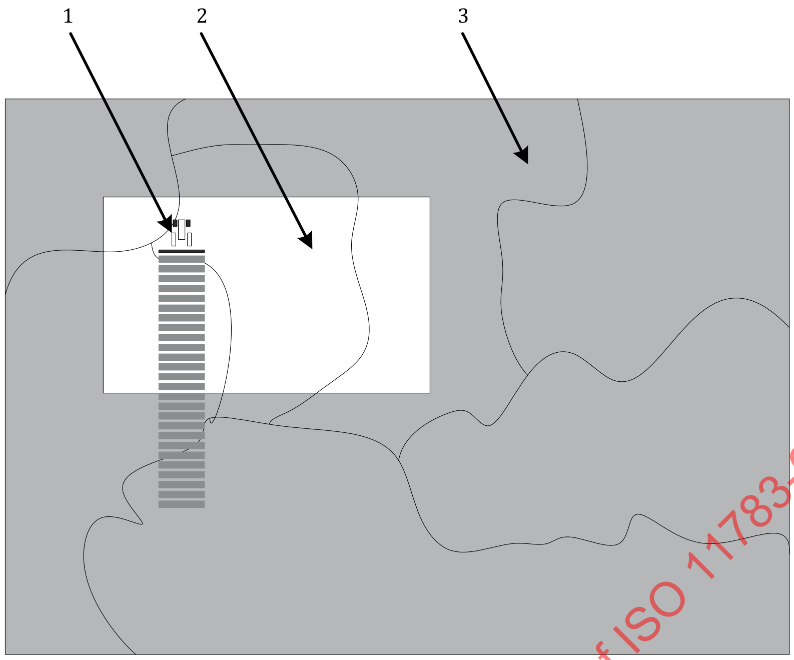
SET FOREGROUND COLOUR [Colour=0 (black)]
SET BACKGROUND COLOUR [Colour=1 (white)]
SET LINE ATTRIBUTES (object id 6019)
SET FILL ATTRIBUTES (NULL Object ID)
SET GRAPHICS CURSOR (X=0, Y=0)
ERASE RECTANGLE (width=30, height=30)
SET GRAPHICS CURSOR (X=0, Y=0)
DRAW LINE (Xoff=0, Yoff=20)
DRAW LINE (Xoff=20, Yoff=0)
MOVE GRAPHICS CURSOR (Xoff=−20, Yoff=0)
SET FOREGROUND COLOUR [Colour=12 (red)]
DRAW LINE (Xoff=6, Yoff=−6)
DRAW LINE (Xoff=8, Yoff=0)
DRAW LINE (Xoff=6, Yoff=−6)
SET GRAPHICS CURSOR (X=3, Y=0)
SET FONT ATTRIBUTES (object id 2000)
DRAW TEXT (Transparent, 13, “Graph Example”)
	

Figure B.16 — Example drawing with Graphics Context object

The Graphics Context object has a transparency option, similar to the Picture Graphic object. Therefore it is possible to create graphics “layers” by overlaying two or more Graphics Context Objects. This allows easy removal or erasure of certain groups of pixels in the object(s). For example, swathe lines could be reset without affected the underlying map image. However, Working Set designers should be cautioned that the Graphics Context object will likely allocate significant memory in the VT (Canvas Width times Canvas Height or more bytes on a 256 colour VT) and should therefore be used sparingly and kept small to avoid rejection of the pool due to memory constraints.



Key

1 picture graphic object

2 viewport

3 graphics context object (entire graphics context referred to as the “canvas”)

Figure B.17 — Example application of the Graphics Context object and viewport

Allowed commands:

— Graphics Context command;

— Change Attribute command;

— Change Background Colour command;

— Get Attribute Value message.

Table B.58 — Graphics Context events

Event	Caused by	VT behaviour	Message
On Change Attribute	Change Attribute command	If field is visible, refresh.	Change Attribute response
On Change Background Colour	Change Background Colour command	Fill the object with the background colour	Change Background Colour response

Table B.59 — Graphics Context attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record Byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=36	3	Object Type = Graphics Context Object (version 4 or later VTs only)
Viewport Width	1	Integer	2	0 — 32767	4—5	Width of the visible viewport in pixels.

Table B.59 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record Byte	Description
Viewport Height	2	Integer	2	0 — 32767	6—7	Height of the visible viewport in pixels.
Viewport X	3	Integer	2	−32768 to +32767	8—9	X Position of the upper left corner of the viewport relative to the upper left corner of the canvas. The viewport is not constrained to the dimensions of the canvas. 0 refers to the left most column of the canvas.
Viewport Y	4	Integer	2	−32768 to +32767	10—11	Y Position of the upper left corner of the viewport relative to the upper left corner of the canvas. The viewport is not constrained to the dimensions of the canvas. 0 refers to the top most row of the canvas.
Canvas Width	[5]	Integer	2	0—32767	12—13	Width of the canvas in pixels.
Canvas Height	[6]	Integer	2	0—32767	14—15	Height of the canvas in pixels.
Viewport Zoom	7	Float	4	−32,0 to +32,0	16—19	Viewport magnification (see Table F.1).
Graphics Cursor X	8	Integer	2	−32768 to +32767	20—21	X Position of the graphics “cursor” relative to the upper left corner of the canvas. Next pixel will be drawn at this location.
Graphics Cursor Y	9	Integer	2	−32768 to +32767	22—23	Y Position of the graphics “cursor” relative to the upper left corner of the canvas. Next pixel will be drawn at this location.
Foreground Colour	10	Integer	1	0—255	24	Foreground colour to use during drawing when options bit 1 is 0.
Background Colour	11	Integer	1	0—255	25	Background colour to use during drawing when options bit 1 is 0. At parsing time, this object is filled with this background colour. NOTE Writing this attribute at runtime fills this object, effectively erasing any content.
Font Attributes Object	12	Integer	2	0—65534, 65535	26—27	Object ID of a Font Attributes Object to use for drawing text. Can be set to NULL if text is not being used.
Line Attributes Object	13	Integer	2	0—65534, 65535	28—29	Object ID of a Line Attributes Object to use for drawing lines and borders or NULL for line suppression.
Fill Attributes Object	14	Integer	2	0—65534, 65535	30—31	Object ID of a Fill Attributes Object to use for filling objects or NULL for no filling.

Table B.59 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record Byte	Description
Format	15	Integer	1	0—2	32	Canvas Type: 0 = Monochrome; 8 pixels per byte. Each bit represents a colour palette index of 0 or 1. ("White" colour can vary with display hardware) 1 = 4 bit colour; 2 colour pixels per byte. Each nibble (4 bits) represents a colour palette index of 0 through 15. 2 = 8 bit colour; 1 colour pixel per byte. Each byte represents a colour palette index of 0 through 255. See Table A.4.
Options	16	Bitmask	1	0—3	33	Bit 0: Transparency 0 = Opaque 1 = Transparent. If opaque, all pixels are drawn in indicated colour. Background objects do not show through. If transparent, pixels in the bitmap that have the transparency colour should show the colour of the background or objects underneath this object instead. Bit 1: Colour 0 = Use Foreground and Background Colours of this object when drawing. 1 = Use Line Colour, Font colour, and Fill Colour, specified in the Line, Font, and Fill attributes when drawing. Bits 2—7 = reserved, set to 0
Transparency Colour	17	Integer	1	0—255	34	Pixels in the bitmap that have this colour index are transparent (background shows through). If opaque, this attribute is ignored.

B.19 Window Mask object

B.19.1 General

The Window Mask object, available in VT version 4 and later, is a parent and special mask object that is used by the VT only in its User-Layout Data Masks. For details, refer to Clause 4.7.1.2.

If the Window Mask Window Type is free form (type 0), the Working Set shall scale this object, just as any other object, to the dimensions of the VT's Window Cell. The aspect ratio of a Window Cell is predictable since there are always 12 window cells (2 columns by 6 rows) on each of the VT's User-Layout Data Masks which, in turn, are always full mask resolution and square. Therefore, the window Cell Size can be calculated from the Data Mask size reported by the VT in the Get Hardware response.

Working Sets can participate in the VT's User-Layout Data Masks, if supported, by placing Window Mask objects in the object pool.

Allowed commands:

— Change Background Colour command;

— Change Child Location command;

— Change Child Position command;

— Change Attribute command;

— Get Attribute Value message.

Table B.60 — Window Mask events

Event	Caused by	VT behaviour	Message
On Show	The User-Layout Data Mask containing this Window Mask object becoming visible on the display.	Fill area with the User-Layout Data Mask background colour. Draw child objects in the order they are listed in the Window Mask object.	VT On User-Layout Hide/Show message with Show indicated
On Hide	The User-Layout Data Mask containing this Window Mask object being removed from the display.		VT On User-Layout Hide/Show message with Hide indicated
On Refresh	Any action that causes a show or hide on a child, grandchild, object, etc.	Redraw objects in the Window Mask that have become corrupted.	—
On Change Background Colour	Change Background Colour command	If the Window Mask is visible and not transparent, fill area with background colour and draw child objects in the order they are listed.	Change Background Colour response
On Change Child Location	Change Child Location command	Draw child object at current location in User-Layout mask background colour to erase it. Refresh Window Mask (to redraw child object or objects).	Change Child Location response
On Change Child Position	Change Child Position command	Draw child object at current location in User-Layout mask background colour to erase it. Refresh Window Mask (to redraw child object or objects).	Change Child Position response
On Change Attribute	Change Attribute command	For behaviour see other change commands above.	Change Attribute response
On Pointing Event press	Operator touches Window mask	Event reported only for Free Form Window	Pointing Event message
On Pointing Event release	Operator touch is released	Event reported only for Free Form Window	Pointing Event message

Table B.61 — Window Mask attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=34	3	Object Type = Window Mask
Width		Integer	1	1—2	4	Width (Number of User-Layout Data Mask columns). Used only if Window Type attribute is 0 (Free Form).

Table B.61 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Height		Integer	1	1—6	5	Height (Number of User-Layout Data Mask rows). Used only if Window Type attribute is 0 (Free Form).
Window Type		Integer	1	0—18	6	Window Type (see B.19.2). The dimensions listed here are width x height in window cells. 0 = Free Form 1 = 1 × 1 Numeric Output Value with Units 2 = 1 × 1 Numeric Output Value, no Units 3 = 1 × 1 String Output Value 4 = 1 × 1 Numeric Input Value with Units 5 = 1 × 1 Numeric Input Value, no Units 6 = 1 × 1 String Input Value 7 = 1 × 1 Horizontal Linear Bar Graph, no Units 8 = 1 × 1 Single Button 9 = 1 × 1 Double Button 10 = 2 × 1 Numeric Output Value with Units

Table B.61 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
						11 = 2 × 1 Numeric Output Value, no Units 12 = 2 × 1 String Output Value 13 = 2 × 1 Numeric Input Value with Units 14 = 2 × 1 Numeric Input Value, no Units 15 = 2 × 1 String Input Value 16 = 2 × 1 Horizontal Linear Bar Graph, no Units 17 = 2 × 1 Single Button 18 = 2 × 1 Double Button
Background Colour	1	Integer	1	0—255	7	Background colour. For Window Type > 0, this attribute shall be ignored by the VT since the VT controls the formatting of those window types.
Options	2	Integer	1	0—3	8	Option bits: Bit 0 = Available. If 0 (FALSE) this window is not available for use at the present time, even though defined. The VT shall not allow the operator to map it and if already mapped, shall blank out the window cell(s) that it occupies. Bit 1 = Transparent. If this bit is 1, the background colour attribute shall not be used and the Window shall be transparent. Bits 2—7 = Reserved, set to 0.
Name	3	Integer	2	0—65534	9—10	Object ID of an Output String object or an Object Pointer object that points to an Output String object that contains the string that gives a proper name to this object. The VT can choose to ignore colour and font information and do its own formatting of the text. The VT shall use this name in its proprietary mapping screen. The VT shall be capable of displaying at least 20 characters.

Table B.61 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Window Title		Integer	2	0—65534, 65535	11—12	Object ID of an Output String object or an Object Pointer object that points to an Output String object that contains the string that supplies window title text. This attribute shall be required for window types above zero. For window type zero, this attribute shall be set to the NULL Object ID. For window types above zero, the VT can choose to ignore colour and font information and do its own formatting of the text.
Window Icon		Integer	2	0—65534, 65535	13—14	Object ID of an output object (as specified in Table A.2 — Allowed hierarchical relationships of objects, “Object Label Graphic Representation” column) that contains an icon for the window. The VT can use this when formatting window types above type zero and can also use it in the proprietary mapping screen to represent the window. This attribute shall only be the NULL if the window type is zero (0). In all other window types, a window icon object shall be supplied.
Number of object references to follow		Integer	1	0—2	15	The number of objects needed in a Window Mask is variable and dependent upon the window type. This attribute, though redundant with the window type attribute, allows for future expansion of this object definition and helps with parsing. If the window type is zero (free form) this attribute shall be set to 0. For all other window types, refer to the tables that follow in Clause B.19.2. Each attribute that follows is a two-byte reference to an object id or the NULL Object ID.
Number of objects to follow		Integer	1	0—255	16	Number of objects to follow even if zero. If the Window Type attribute is not zero, this attribute shall be set to 0. Child objects are only necessary if the Window Type is Free Form (0). Each of these objects is “contained” in this object. Each object consists of 6 bytes: two (2) for Object ID and four (4) for location.

Table B.61 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Number of macros to follow		Integer	1	0—255	17	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Object ID}		Integer	2	0—65535	18 + referenced object index*2	Object ID of a required object reference for the given window type (see B.19.2). List all objects before listing macros.
Repeat: {Object ID}		Integer	2	0—65534	18 + num referenced objects*2 + child object index*6	Object ID of an object contained in this object (see A.1.3 Object relationships). List all objects before listing macros.
{X Location}		Signed integer	2	−32768 to +32767	20 + num referenced objects*2 + child object index*6	Relative X location of the top left corner of the object (relative to the top left corner of the Window Mask object).
{Y Location}		Signed integer	2	−32768 to +32767	22 + num referenced objects*2 + child object index*6	Relative Y location of the top left corner of the object (relative to the top left corner of the Window Mask object).
Repeat: {Event ID}		Integer	1	0—255	18 + num referenced objects*2 + num child objects*6 + macro index*2	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3)

Table B.61 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
{Macro ID}		Integer	1	0—255	19 + num referenced objects*2 + num child objects*6 + macro index*2	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)

B.19.2 Window Mask Window Types

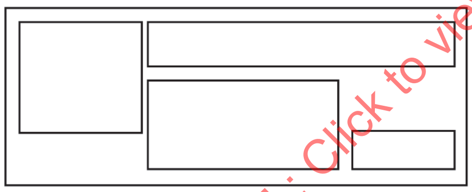
The Window Mask Type attribute in the Window Mask, along with the object component references in the object allow the VT to create a uniform look and feel for standardized windows from all Working Sets. In addition, when the attribute is zero (0 = Free Form) the Working Set can create completely custom presentation.

For the Window Masks (excluding Free Form) which contain a Linear Bar Graph object, the VT cannot assume the minimum value and maximum value attributes of the Linear Bar Graph represent displayable units of measure; therefore the VT should not label the scale on the bar graph.

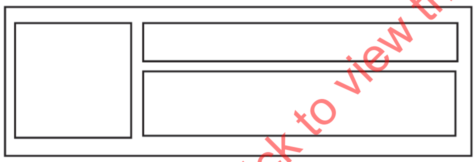
B.19.2.1 Free Form Window Mask (type 0)

When the attribute is 0, the Working Set supplies and positions all child objects contained inside the window. In this case the Working Set has complete control over the look and feel of the window and the VT shall render the Window Mask exactly as specified by the child objects, position, size, and transparency attributes of the Window Mask object and all other formatting attributes.

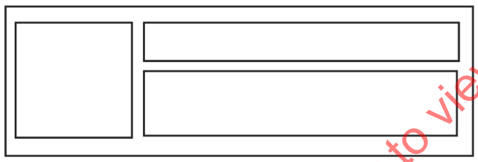
B.19.2.2 1 × 1 Numeric Output Value Window with Units

Window Type	1
Description	This window displays a single numeric output with units of measure in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric value and units of measure
Number of Object References	2
Object References (in order)	#1 — Output Number (for the numeric value) #2 — Output String (for the units of measure)
VT Field Lengths	Window Title: 11 characters Window Value: 5 characters (including decimal place if needed) Window Units: 5 characters
VT Formatting and Scaling	The VT can format the window as desired, and can ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT can scale objects, excluding the Window Icon, if required or desired.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3.
Example Layout	 This figure is an example that shows all components of the window. VT designs control the formatting and layout of this window.

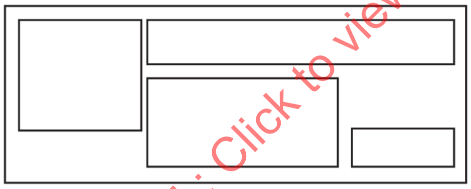
B.19.2.3 1 × 1 Numeric Output Value Window, No Units

Window Type	2
Description	This window displays a single numeric output with no units of measure in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric value
Number of Object References	1
Object Reference	#1 — Output Number (for the numeric value)
VT Field Lengths	Window Title: 11 characters Window Value: 11 characters (including decimal place if needed)
VT Formatting and Scaling	The VT can format the window as desired, and can ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT can scale objects, excluding the Window Icon, if required or desired.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3.
Example Layout	 This figure is an example that shows all components of the window. VT designs control the formatting and layout of this window.

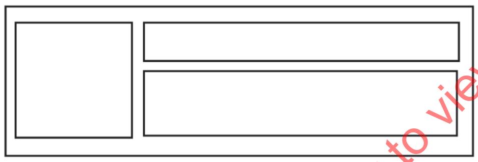
B.19.2.4 1 × 1 String Output Value Window

Window Type	3
Description	This window displays a single string output in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	String Value
Number of Object References	1
Object Reference	#1 — Output String (for the string value)
VT Field Lengths	Window Title: 11 characters Window Value: 11 characters
VT Formatting and Scaling	The VT can format the window as desired, and can ignore colour and Font Attributes in the referenced object. Field length shall conform to the above requirements. The VT can scale objects, excluding the Window Icon, if required or desired.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3.
Example Layout	 This figure is an example that shows all components of the window. VT designs control the formatting and layout of this window.

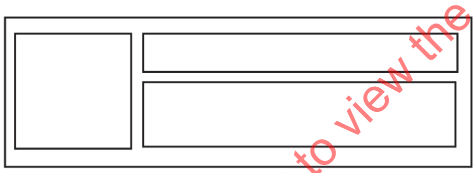
B.19.2.5 1 × 1 Numeric Input Value Window with Units

Window Type	4
Description	This window displays a single numeric input with units of measure in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric value and units of measure
Number of Object References	2
Object References (in order)	#1 — Input Number (for the numeric value) #2 — Output String (for the units of measure)
VT Field Lengths	Window Title: 11 characters Window Value: 5 characters (including decimal place if needed) Window Units: 5 characters
VT Formatting and Scaling	The VT can format the window as desired, and can ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT can scale objects, excluding the Window Icon, if required or desired.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3.
Example Layout	 This figure is an example that shows all components of the window. VT designs control the formatting and layout of this window.

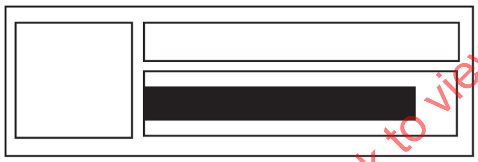
B.19.2.6 1 × 1 Numeric Input Value Window, No Units

Window Type	5
Description	This window displays a single numeric input with no units of measure in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric value
Number of Object References	1
Object Reference	#1 — Input Number (for the numeric value)
VT Field Lengths	Window Title: 11 characters Window Value: 11 characters (including decimal place if needed)
VT Formatting and Scaling	The VT can format the window as desired, and can ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT can scale objects, excluding the Window Icon, if required or desired.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3.
Example Layout	 This figure is an example that shows all components of the window. VT designs control the formatting and layout of this window.

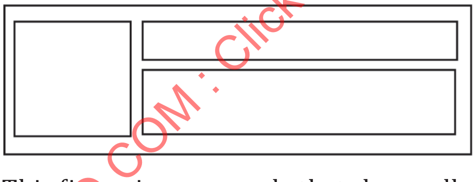
B.19.2.7 1 × 1 String Input Value Window

Window Type	6
Description	This window displays a single string input in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	String Value
Number of Object References	1
Object Reference	#1 — Input String object (for the string value)
VT Field Lengths	Window Title: 11 characters Window Value: 11 characters
VT Formatting and Scaling	The VT can format the window as desired, and can ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT can scale objects, excluding the Window Icon, if required or desired.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3.
Example Layout	 This figure is an example that shows all components of the window. VT designs control the formatting and layout of this window.

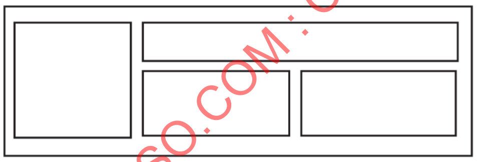
B.19.2.8 1 × 1 Horizontal Linear Bargraph Window

Window Type	7
Description	This window displays a single horizontal linear bargraph in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Bar Graph
Number of Object References	1
Object Reference	#1 — Linear Bar Graph
VT Field Lengths	Window Title: 11 characters
VT Formatting and Scaling	The VT can format the window as desired, and can ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT can scale objects, excluding the Window Icon, if required or desired.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3. The Working Set shall supply an Output Linear Bar Graph object in the horizontal position. The bar graph should increase from left to right but can increase in either direction.
Example Layout	 This figure is an example that shows all components of the window. VT designs control the formatting and layout of this window.

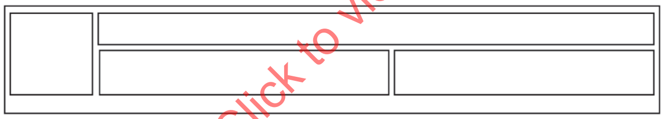
B.19.2.9 1 × 1 Single Button Window

Window Type	8
Description	This window displays a single Button object in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	One Button
Number of Object References	1
Object Reference	#1 — Button
VT Field Lengths	Window Title: 11 characters
VT Formatting and Scaling	The VT design is free to format the window as desired, and can ignore colour and Font Attributes in the referenced Button object. Field length requirements above shall be met. The VT is free to scale objects, excluding the Window Icon, if needed or desired but in the case of the Button object shall only maintain or increase it's size to avoid clipping child objects.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3. The Working Set shall also scale the Button object and its children according to these equations: Button Width = Window Cell Width × 65 % (rounded down) Button Height = Window Cell Height × 57 % (rounded down) Child objects shall be similarly scaled. Due to these requirements, the same Button object in a Data Mask cannot be used in a Window Mask since the scale factors are different.
Example Layout	 This figure is an example that shows all components of the window. VT designs control the formatting and layout of this window.

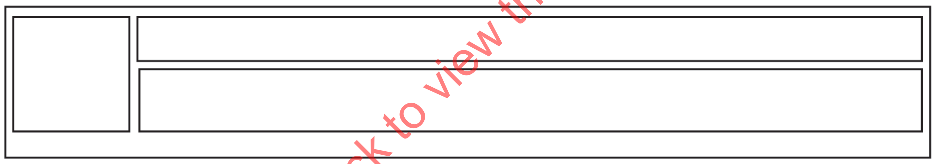
B.19.2.10 1 × 1 Double Button Window

Window Type	9
Description	This window displays two Button objects in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Two Buttons
Number of Object References	2
Object References (in order)	#1 — Button (for the left side button) #2 — Button (for the right side button)
VT Field Lengths	Window Title: 11 characters
VT Formatting and Scaling	The VT design is free to format the window as desired, and can ignore colour and Font Attributes in the referenced Button objects. Field length requirements above shall be met. The VT is free to scale objects, excluding the Window Icon, if needed or desired but in the case of the Button objects shall only maintain or increase their size to avoid clipping child objects.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3. The Working Set shall also scale the Button objects and their children according to these equations: Button Width = Window Cell Width x 30 % (rounded down) Button Height = Window Cell Height x 57 % (rounded down) Child objects shall also be similarly scaled. Due to these requirements, the same Button object in a Data Mask cannot be used in a Window Mask since the scale factors are different.
Example Layout	 This figure is an example that shows all components of the window. VT designs control the formatting and layout of this window.

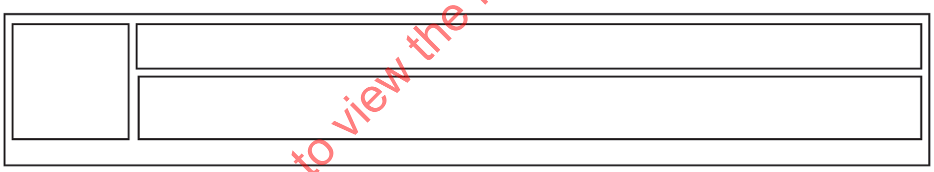
B.19.2.11 2 × 1 Numeric Output Value Window with Units

Window Type	10
Description	This window displays a single numeric output with units of measure in two horizontal window cells.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric value and units of measure
Number of Object References	2
Object References (in order)	#1 — Output Number (for the numeric value) #2 — Output String (for the units of measure)
VT Field Lengths	Window Title: 20 characters Window Value: 10 characters (including decimal place if needed) Window Units: 9 characters
VT Formatting and Scaling	The VT can format the window as desired, and can ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT can scale objects, excluding the Window Icon, if required or desired.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3.
Example Layout	 This figure is provided for example only and shows all components of the window. VT designs control the formatting and layout of this window.

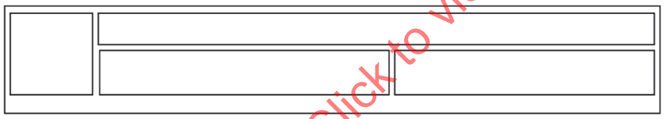
B.19.2.12 2 × 1 Numeric Output Value Window, No Units

Window Type	11
Description	This window displays a single numeric output with no units of measure in two horizontal window cells.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric value
Number of Object References	1
Object Reference	#1 — Output Number (for the numeric value)
VT Field Lengths	Window Title: 20 characters Window Value: 20 characters (including decimal place if needed)
VT Formatting and Scaling	The VT can format the window as desired, and can ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT can scale objects, excluding the Window Icon, if required or desired.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3.
Example Layout	 This figure is provided for example only and shows all components of the window. VT designs control the formatting and layout of this window.

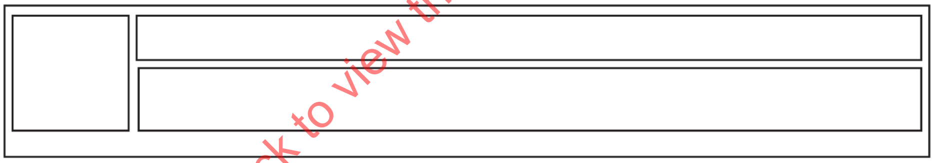
B.19.2.13 2 × 1 String Output Value Window

Window Type	12
Description	This window displays a single string output in two horizontal window cells.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	String Value
Number of Object References	1
Object Reference	#1 — Output String (for the string value)
VT Field Lengths	Window Title: 20 characters Window Value: 20 characters
VT Formatting and Scaling	The VT can format the window as desired, and can ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT can scale objects, excluding the Window Icon, if required or desired.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3.
Example Layout	 This figure is provided for example only and shows all components of the window. VT designs control the formatting and layout of this window.

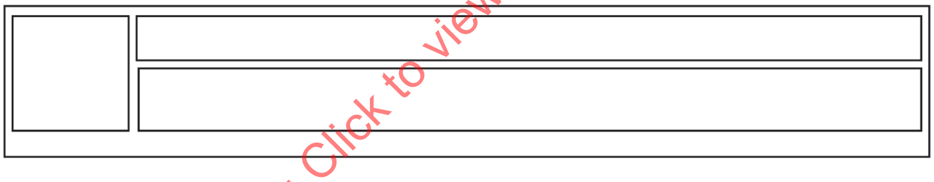
B.19.2.14 2 × 1 Numeric Input Value Window with Units

Window Type	13
Description	This window displays a single numeric input with units of measure in two horizontal window cells.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric value and units of measure
Number of Object References	2
Object References (in order)	#1 — Input Number (for the numeric value) #2 — Output String (for the units of measure)
VT Field Lengths	Window Title: 20 characters Window Value: 10 characters (including decimal place if needed) Window Units: 9 characters
VT Formatting and Scaling	The VT can format the window as desired, and can ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT can scale objects, excluding the Window Icon, if required or desired.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3.
Example Layout	 This figure is provided for example only and shows all components of the window. VT designs control the formatting and layout of this window.

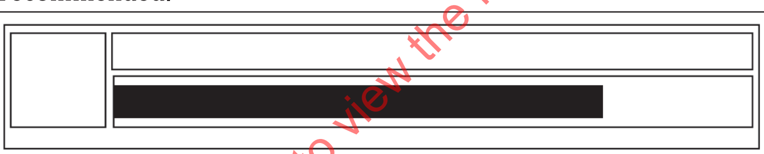
B.19.2.15 2 × 1 Numeric Input Value Window, No Units

Window Type	14
Description	This window displays a single numeric input with no units of measure in two horizontal window cells.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric Value
Number of Object References	1
Object Reference	#1 — Input Number (for the numeric value)
VT Field Lengths	Window Title: 20 characters Window Value: 20 characters (including decimal place if needed)
VT Formatting and Scaling	The VT can format the window as desired, and can ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT can scale objects, excluding the Window Icon, if required or desired.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3.
Example Layout	 This figure is provided for example only and shows all components of the window. VT designs control the formatting and layout of this window.

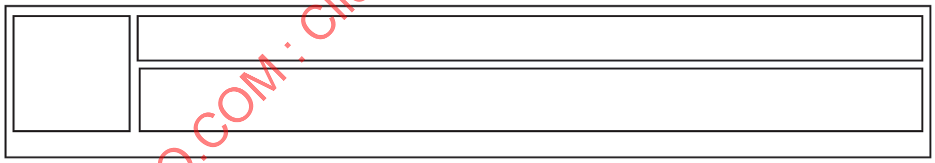
B.19.2.16 2 × 1 String Input Value Window

Window Type	15
Description	This window displays a single string input in two horizontal window cells.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	String Value
Window Value Type	Input String object
Number of Object References	1
Object Reference	#1 — Input String object (for the string value)
VT Field Lengths	Window Title: 20 characters Window Value: 20 characters
VT Formatting and Scaling	The VT can format the window as desired, and can ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT can scale objects, excluding the Window Icon, if required or desired.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3.
Example Layout	 This figure is provided for example only and shows all components of the window. VT designs control the formatting and layout of this window.

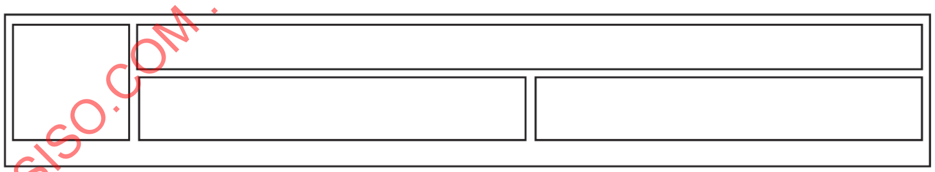
B.19.2.17 2 × 1 Horizontal Linear Bargraph Window

Window Type	16
Description	This window displays a single horizontal linear bargraph in two horizontal window cells.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Bar Value
Number of Object References	1
Object Reference	#1 — Linear Bar Graph
VT Field Lengths	Window Title: 20 characters
VT Formatting and Scaling	The VT can format the window as desired, and can ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT can scale objects, excluding the Window Icon, if required or desired.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3. The Working Set shall supply a linear bargraph object in the horizontal position. Bargraph can grow in either direction but left to right is recommended.
Example Layout	 This figure is provided for example only and shows all components of the window. VT designs control the formatting and layout of this window.

B.19.2.18 2 × 1 Single Button Window

Window Type	17
Description	This window displays a single Button object in two horizontal window cells.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	One Button
Number of Object References	1
Object Reference	#1 — Button
VT Field Lengths	Window Title: 20 characters
VT Formatting and Scaling	The VT design is free to format the window as desired, and can ignore colour and Font Attributes in the referenced Button object. Field length requirements above shall be met. The VT is free to scale objects, excluding the Window Icon, if needed or desired but in the case of the Button object shall only maintain or increase its size to avoid clipping child objects.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3. The Working Set shall also scale the Button object and its children according to these equations: Button Width = Window Cell Width × 80 % (rounded down) Button Height = Window Cell Height × 57 % (rounded down) Child objects shall also be scaled accordingly. Due to these rules, it is likely not possible to use the same Button object in a Data Mask and a Window Mask since the scale factors are different.
Example Layout	 This figure is provided for example only and shows all components of the window. VT designs control the formatting and layout of this window.

B.19.2.19 2 × 1 Double Button Window

Window Type	18
Description	This window displays two Button objects in two horizontal window cells.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Two Buttons
Number of Object References	2
Object References (in order)	#1 — Button (for the left side button) #2 — Button (for the right side button)
VT Field Lengths	Window Title: 20 characters
VT Formatting and Scaling	The VT design is free to format the window as desired, and can ignore colour and Font Attributes in the referenced Button objects. Field length requirements above shall be met. The VT is free to scale objects, excluding the Window Icon, if needed or desired but in the case of the Button objects shall only maintain or increase their size to avoid clipping child objects.
Working Set Formatting and Scaling	The Working Set has no control over formatting or layout. The Working Set shall scale the Window Icon object according to 4.7.15.3. The Working Set shall also scale the Button objects and their children according to these equations: Button Width = Window Cell Width × 40 % (rounded down) Button Height = Window Cell Height × 57 % (rounded down) Child objects shall also be scaled accordingly. Due to these rules, it is likely not possible to use the same Button object in a Data Mask and a Window Mask since the scale factors are different.
Example Layout	 This figure is provided for example only and shows all components of the window. VT designs control the formatting and layout of this window.

B.20 Key Group object

The Key Group object, available in VT version 4 and later, is a parent object that is used by the VT only in its User-Layout Soft Key Mask. For details, refer to 4.7.8. The Key Group object contains Key objects and Object Pointer objects. An Object Pointer object shall point only to a Key object, another Object Pointer object, or the NULL object.

The Key objects contained in this object shall be a grouping of Key objects, or Object Pointers to Key objects. The VT shall not allow the operator to break up, even across visible Soft Key page boundaries, this grouping when mapping the Key layouts and shall require the operator to map the entire group together. This also means that several Key Cells can be required to represent this object.

Working Sets can participate in the VT's User-Layout Soft Key Mask, if supported, by placing Key Group objects in the object pool.

Working Set designers should make the Key Group object transparent. This allows the VT to set the background colour of each child Key object so that all Keys have the same background colour. If required, the Working Set can determine the VT's background colour using the Get Window Mask Data message.

Object Pointer objects pointing to the NULL Object ID reserve a Key object position (the remaining Key objects do not move up and the trailing Key object can be navigated to.

Allowed commands:

— Change Attribute command;

— Get Attribute Value message.

Table B.62 — Key Group events

Event	Caused by	VT behaviour	Message
On Change Attribute	Change Attribute command	Modify attributes of this object.	Change Attribute response

Table B.63 — Key Group attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=35	3	Object type = Key Group
Options	1	Integer	1	0—3	4	Option bits: Bit 0 = Available. If 0 (FALSE) this object is not available for use at the present time, even though defined. The VT shall not allow the operator to map it and if already mapped, shall blank out the key cell(s) that it occupies. Bit 1 = Transparent. If this bit is 1, the VT shall ignore the background colour attribute in all child Key objects and shall set the background colour as desired. Bits 2—7 = Reserved, set to 0.
Name	2	Integer	2	0—65534	5—6	Object ID of an Output String object or an Object Pointer object that points to an Output String object that contains the string that gives a proper name to this object. The VT can choose to ignore colour and font information and do its own formatting of the text. The VT shall use this name in its proprietary mapping screen. The VT shall be capable of displaying at least 20 characters.

Table B.63 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Key Group Icon		Integer	2	0—65534, 65535	7—8	Object ID of an output object (as specified in Table A.2, “Object Label Graphic Representation” column) that contains an optional icon for the key group. The VT can use this in the proprietary mapping screen to represent the key group. The representation of the Key Group is VT proprietary if the NULL Object ID is used. It is recommended not to use a transparent background for objects as these objects can appear on a proprietary mapping screen where the background colour is unknown by the designer.
Number of objects to follow		Integer	1	1—4	9	Number of Key or Object Pointer objects to follow. After dereferencing pointers, there shall be a maximum of 4 Key objects per Key Group object.
Number of macros to follow		Integer	1	0—255	10	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Object ID}		Integer	2	0—65534	11 + object*2	Object ID of an object contained in this Key Group. (See A.1.3). List all objects before listing macros.
Repeat: {Event ID}		Integer	1	0—255	11 + (No. objects*2)...	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3)
{Macro ID}		Integer	1	0—255	12 + (No. objects*2)...	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)

B.21 Object Label Reference List object

The Object Label Reference List object, available in VT version 4 and later, provides a mechanism to assign a String Variable and/or a graphical designator as a label to other objects. An Object Pool shall not contain more than one Object Label Reference List object.

An object label is only intended for use by the VT in various proprietary screens and popup messages or editors, and recommended for use in new Working Set designs. Object Labels are useful in two specific cases, though not limited to these two:

— It is recommended that Working Set designs provide an object label with a text name for the Working Set object. This text can be used by the VT to identify the Working Set in proprietary screens and alarms (e.g. "Planter" could be used to differentiate one Working Set from others).

— It is recommended that Working Set designs provide an Object Label for all input objects. Since popup editor windows in the VT can cover the focused input object, it is also recommended that VT designs display the Object Label in the popup editor window so that the operator can recall what is being edited. An example of such a label could be "Section 1 Width (meters)" and/or an appropriate designator graphic.

Since the object label is intended for proprietary screens, any runtime changes to an already visible object label might not cause an update to the existing presentation. The changes should be seen the next time the object label appears (e.g. popup editor window is closed and is later reopened).

Strings in excess of 32 characters can be clipped to 32 characters by the VT. Referenced designator graphics shall fit within a Soft Key designator area (see 4.5.3 Soft Key Mask area and Soft Key designators). The referenced designator object can contain objects as listed in Object Label graphic representation (see Table A.2). It is not possible to assign more than one label to an object. When an Object ID, of an object to label, appears more than one time in the Object Label Reference List, the Object Pool shall be rejected. When an object is used as a label, and this object also has a label, this latter label shall not be displayed.

To permit disabling an Object Label, both the String Variable reference and the Graphical Reference could be set to NULL and the VT should be designed to not show an Object Label.

Allowed commands:

— Change Object Label command;

— Get Attribute Value message.

Table B.64 — Object Label Reference List attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=40	3	Object Type = Object Label Reference List
Number of La-belled objects	[1]	Integer	2	0—65535	4—5	Number of labelled objects to follow. One labelled object consumes 7 bytes.
Repeat: {Object ID}		Integer	2	0—65534	6—7...	Object ID of object to label.
{String Varia-ble reference}		Integer	2	0—65535	8—9...	Object ID of a String Variable object that contains the label string or FFFF$_{16}$ if no text is supplied
{Font type}		Integer	1	0—255	10...	Font type (see Annex K) (ignored if String Variable object reference is NULL or the string contains a WideString (see 4.6.19.7).
{Object Label graphic rep-resentation}		Integer	2	0—65535	11—12...	Object ID of an object to be used as a graphic representation of the object label or FFFF$_{16}$ if no designator supplied. When the VT draws this object it shall be clipped to the size of a Soft Key designator. See Table A.2.

B.22 External Object Definition object

The External Object Definition object, available in VT version 5 and later, lists the objects which another WS is allowed to reference through the External Object Pointer.

When an object pool is loaded from non-volatile memory the VT shall clear the Enable bit in the Options attribute of all External Object Definition objects. See 4.6.11.6. Table B.65 and Table B.66.

Allowed commands:

— Change Attribute command;

— Change List Item command;

— Get Attribute Value message.

Table B.65 — External Object Definition events

Event	Caused by	VT behaviour	Message
On Change Attribute	Change Attribute command	Reevaluate all currently displayed External Object Pointer objects which reference objects in the WS which owns this object.	Change Attribute response

Table B.66 — External Object Definition attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=41	3	Object Type = External Object Definition
Options	1	Integer	1	0—1	4	Logical bits to indicate options. 1 = TRUE. Bit 0 = Enabled. If TRUE the object is enabled and the WS identified by the NAME attributes is allowed to reference objects via the External Object Pointer object. If FALSE this object is disabled and shall be ignored.
NAME 0	2[a]	Integer	4	0 to 2^32—1	5 — 8	Byte 1—4 of the NAME of the WS Master of the WS which shall be allowed to reference the objects in the object list.
NAME 1	3[a]	Integer	4	0 to 2^32—1	9—12	Byte 5—8 of the NAME of the WS Master of the WS which shall be allowed to reference the objects in the object list.
Number of object to follow		Integer	1	0—255	13	Number of objects to follow even if zero. Each object consists of 2 bytes.
Repeat: {Object ID}		Integer	2	0—65534, 65535	14 + object*2	These objects make up the list of objects which can be externally referenced. NULL is a "no item" placeholder. The Change List Item command allows objects to be replaced.

[a] It is recommended that the WS clears the Enabled bit in the Options attribute before updating the NAME attributes.

B.23 External Reference NAME object

The External Reference NAME object, available in VT version 5 and later, identifies the WS master of a WS which can be referenced through the External Object Pointer.

When an object pool is loaded from non-volatile memory the VT shall clear the Enable bit in the Options attribute of all External Reference NAME objects.

See Table B.67 and Table B.68.

Allowed commands:

— Change Attribute command;

— Get Attribute Value message.

Table B.67 — External Reference NAME events

Event	Caused by	VT behaviour	Message
On Change Attribute	Change Attribute command	Reevaluate all currently displayed External Object Pointer objects which reference this object.	Change Attribute response

Table B.68 — External Reference NAME attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=42	3	Object Type = External Reference NAME
Options	1	Integer	1	0—1	4	Logical bits to indicate options. 1 = TRUE. Bit 0 = Enabled. If TRUE the object is enabled and the WS identified by the NAME attributes can be referenced by an External Object Pointer object. If FALSE the object is disabled and any External Object Pointer object referencing this object shall be considered invalid.
NAME 0	2[a]	Integer	4	0 to 2^32—1	5—8	Byte 1—4 of the NAME of the referenced WS Master.
NAME 1	3[a]	Integer	4	0 to 2^32—1	9—12	Byte 5—8 of the NAME of the referenced WS Master.

[a] It is recommended that the WS clears the Enabled bit in the Options attribute before updating the NAME attributes.

B.24 External Object Pointer object

The External Object Pointer object, available in VT version 5 and later, allows a Working Set to display objects that exist in another Working Set's object pool. By changing the value of the pointer object, a different object can be referenced to the same location. An External Object Pointer can point to the NULL Object ID and in this case the Default Object shall be drawn.

The Default Object is drawn under the following conditions:

— The External Object Pointer, or any directly followed Object Pointer, points to the NULL Object ID;

— If any child object of the External Object Pointer is invalid based on the object hierarchy (see Table A.2).

When an object pool is loaded from non-volatile memory the VT shall set the External Object Id attribute of all External Object Pointer objects to the NULL object id. This forces the WS to renew the External Object ID.

It is intended that any object or object hierarchy referenced by an External Object Pointer object is still owned and controlled by the referenced parent Working Set and Object Pool.

When objects from a Working Set's object pool are referenced by another Working Set via the External Object Pointer, defined events and macros shall execute on these referenced objects and shall do so in the context of the original parent object pool. For example, if a referenced Button object has a macro object associated with the On Key Press, the associated macro object is executed in the context of the original parent object pool. Any messages resulting from the execution of the macro would be sent to the original parent Working Set, not the referencing Working Set. Any additional events caused by the executing macro shall also be executed in the context of the original parent Object Pool.

When objects from a Working Set's object pool are referenced by another Working Set via the External Object Pointer, and when the original parent Working Set sends commands that act on or change attributes of the referenced objects, these objects shall be refreshed in the displayed, referencing, mask or parent object. The current Colour Map and Colour Palette objects, if any, in effect in the original parent Object Pool shall be referenced for colours. In other words, all attributes used for drawing shall come from the context of the original parent Working Set's object pool. It shall not be permissible for the referencing Working Set to send commands or create macros that act directly upon objects referenced from another Working Set.

With respect to objects in another object pool that are referenced by an External Object Pointer object, all messages specifically associated with these object(s) shall be sent by the VT to the original parent Working Set, not the referencing Working Set. For example, all navigation and data input related messages that are associated with the referenced objects shall be sent to the original parent Working Set. In these cases, whenever a "Parent ID" or "Parent Mask" or similar attribute is required in the message, the VT shall use the Object ID of the original parent Object Pool's External Object Definition object. In the case of multiple External Object Definition objects assigned to the NAME of the referencing Working Set containing the Object ID of the referenced object, it is proprietary to the VT which of these it reports as parent object. This is special notification to the Working Set receiving the message, that the event has occurred in the context of another referencing Working Set, even though the receiving Working Set is not the active Working Set.

See Table B.69 and Table B.70.

Allowed commands:

— Change Numeric Value command;

— Change Attribute command;

— Get Attribute Value message.

Table B.69 — External Object Pointer events

Event	Caused by	VT behaviour	Message
On Change Attribute	Change Attribute command	Reevaluate the External Object Pointer object and redraw if required.	Change Attribute response

Table B.70 — External Object Pointer attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=43	3	Object Type = External Object Pointer
Default Object ID	1	Integer	2	0—65534, 65535	4—5	Object ID of an object which shall be displayed if the External Object ID is not valid, or the NULL Object ID. NOTE The default object is always from the same object pool as this object.
External Reference NAME ID	2	Integer	2	0—65534, 65535	6—7	Object id of an External Reference NAME object or the NULL Object ID.
External Object ID	3	Integer	2	0—65534, 65535	8—9	Object ID of a referenced object or the NULL Object ID. The referenced object is found in the object pool of the Working Set Master identified by the External Reference NAME ID attribute and listed in the corresponding External Object Definition object.

B.25 Animation object

The Animation object, available in VT version 5 and later, is used to display animations. This object consists of a list of object ids, one of which is drawn determined by the Value attribute.

Working Set designers should be aware that the performance of the VT cannot be predicted, so the size of the individual objects should be small and the refresh rate kept reasonable (200 ms or slower is recommended). The refresh rate attribute is a suggestion only and cannot be guaranteed by the VT design. The VT can limit or modify the refresh interval.

The VT shall increment the index Value when this object is enabled and the refresh interval is non-zero and is a visible member of an active mask and the specified Refresh Interval has expired. When not visible, the animation timer is suspended, so the index Value shall not be incremented. If multiple instances of this object are visible, the same referenced object shall be visible in each instance. Further, the Refresh Interval is based on the object, and shall not be affected by the number of visible instances.

When the VT prepares to increment the index Value, it shall be range checked against the “First Child Index” and the “Last Child Index”. If the index Value<First Child index, it shall be set to the value of the First Child Index and then incremented (in this case, the object referenced by the First Child Index is not shown). If the index Value>Last Child Index, it shall be set to Last Child Index and then the behaviour is controlled by the Animation Sequence value in the Options attribute.

The VT shall not display anything for the selected item in the following cases:

— index value is 255 which means “no item is chosen”;

— index value is invalid (greater than the number of items in the list minus 1);

— selected list item is a no-item placeholder (NULL);

— selected list item is an Object Pointer with a value of NULL;

— selected list item is a Container, and the Container is in the hidden state;

— the object is enabled and any of the following numeric relationships are not true:

  — 0 ≤ First Child Index ≤ Last Child Index;

— 0 ≤ Last Child Index < number of items in the list;

— 0 ≤ Default Child Index < number of items in the list.

Note that this object can contain more child objects than are defined by the range "First Child Index" to "Last Child Index". This allows the working set to change the animation by changing the values of the "First Child Index" and "Last Child Index" attributes without the need to upload a new child object list.

The Animation object animation sequence can be configured in either Single Shot or Loop modes.

— Single Shot mode: In Single Shot mode, the animation sequence is played only once starting with the List Index being initialized to the "First Child Index" value, and ending with the List Index being equal to the "Last Child Index" value. Once complete, the animation stops and the last child object remains displayed as long as the object is enabled. The single-shot animation sequence can be altered or replayed by changing the index Value or by disabling and then re-enabling the object which resets the index Value to the "First Child Index". If this object is initially enabled in the object pool, then the animation is played starting with the child referenced by the Index value the first time this object is displayed.

— Loop mode: In Loop mode, the animation is repeated as long as the object is enabled. After the child object at the "Last Child Index" is displayed, the index Value is reset to the "First Child Index" attribute value and the child object at that index is displayed. This cycle repeats until the object is disabled.

If the Working Set changes the index Value at runtime, the current refresh interval is restarted and the selected object is drawn.

When the Animation object is disabled, the presentation is defined by one of four modes.

— Pause mode: The index Value is unchanged and the child at this index is shown. This mode allows animations to be enabled and disabled without skipping child objects in the animation.

— Reset to First mode: The index Value is reset to the "First Child Index" when disabled, and the child at this index is drawn.

— Default Object mode: The index Value is unchanged, however the child at the "Default Child Index" is drawn.

— Blank mode: The index Value is unchanged, and no object is drawn.

Allowed commands:

— Enable/Disable Object command;

— Change Numeric Value command;

— Change Attribute command;

— Change List Item command;

— Change Size command;

— Get Attribute Value message.

Table B.71 — Animation events

Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Redraw this object.	—
On Enable	Enable/Disable Object command	Mark the object enabled. Set Enabled attribute to 1 (animating)	Enable/Disable Object Response
On Disable	Enable/Disable Object command	Mark the object disabled. Set Enabled attribute to 0 (stopped, behaviour according to the Disabled Behaviour option)	Enable/Disable Object Response
On Change Value	Change Numeric Value command (to change the list index)	If object is displayed, redraw object with new value. Refresh parent object.	Change Numeric Value response
On Change Attribute	Change Attribute command	For behaviour, see Change Background Colour command.	Change Attribute response
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size response

Table B.72 — Animation attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=44	3	Object Type = Animation object
Width	1	Integer	2	0—65535	4—5	Maximum width of the Animation object's area in pixels. Objects or portions of objects outside the defined area are clipped.
Height	2	Integer	2	0—65535	6—7	Maximum height of the Animation object's area in pixels. Objects or portions of objects outside the defined area are clipped.
Refresh Interval	3	Integer	2	0—65535	8—9	Desired time in ms between refreshes of this object. The value zero stops the timer but is not equivalent to enabled = 0.
Value	4	Integer	1	0—254, 255	10	List Index of the object to be shown. The first item is at index zero (0).
Enabled	5	Integer	1	0 or 1	11	When set to 1, this object is enabled (animating). When set to 0, this object is disabled (stopped).
First Child Index	6	Integer	1	0—254	12	This attribute represents the index of the first child object in the animation sequence.
Last Child Index	7	Integer	1	0—254	13	This attribute represents the index of the last child object in the animation sequence.
Default Child Index	8	Integer	1	0—254	14	This attribute represents the index of the default child object in the animation sequence. See Options attribute, Disabled Behaviour.

Table B.72 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Options	9	Integer	1	0–7	15	Options: Bit 0 = Animation Sequence 0 = Single Shot mode 1 = Loop mode Bits 1–2 = Disabled Behaviour 0 = Pause mode 1 = Reset to First mode 2 = Default Object mode 3 = Blank mode Bits 3–7 = 0 reserved
Number of objects to follow		Integer	1	0–255	16	Number of object references to follow even if zero. Each object consists of 6 bytes: two (2) for object ID and four (4) for location.
Number of macros to follow		Integer	1	0–255	17	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Object ID}		Integer	2	0–65534	18 + object*6	Object ID of an object in this animation object (see A.1.3). List all objects before listing macros.
{X Location}		Signed integer	2	−32768 to +32767	20 + object*6	Relative X location of the top left corner of the object (relative to the top left corner of the Animation object).
{Y Location}		Signed integer	2	−32768 to +32767	22 + object*6	Relative Y location of the top left corner of the object (relative to the top left corner of the Animation object).
Repeat: {Event ID}		Integer	1	0–255	18 + (No. objects *6)…	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3)
{Macro ID}		Integer	1	0–255	19 + (No. objects *6)…	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3)

B.26 Colour Palette object

The Colour Palette object, available in VT version 6 and later, is used to replace the VT standard colour palette in-use for this Working Set. This object contains up to 256 ARGB values arranged in the order of the VT colour number. This allows the working set to fully define the colours to be used.

The object pool can contain zero or more Colour Palette objects. After the pool is loaded the VT uses the colour palette defined by the Working Set Special Controls object. Every object of a Working Set's pool shall be displayed with that Working Set's Colour Palette object. Note that the Colour Map object can also be active in the process of resolving the displayed colour (see 4.8 and 4.6.13).

The Colour Palette object contains the definitions for a specified number of valid colour index values, as defined by the “Number of ARGB values to follow” attribute. With this, a subset of the entire colour palette can be redefined, starting with colour zero. If an object pool update is processed which contains a replacement Colour Palette object, the VT will first initialize the colour palette for the Working Set to the VT standard colour palette and then update that colour palette based on the new object. Each colour index is represented by 4 bytes for the ARGB value.

The inclusion of the Alpha channel permits enhanced presentation, however the designer should avoid transparency for any object(s) that can be overlaid on another object where the designer does not have control of the underlying objects colour (e.g. transparency of a Data Mask background colour is not recommended as the colour beneath the Data Mask is unknown to the object pool designer. Transparency on a Button object is not recommended if it can be referenced from an External Object Pointer object).

The ARGB value is then placed in little-endian order resulting in the byte sequence of B, G, R, and A, as seen in Table B.73.

Allowed commands:

— Change Attribute command;

— Get Attribute Value message.

Table B.73 — Colour Palette attributes and record format

Attribute Name	**AID**	**Type**	**Size (bytes)**	**Range or Value**	**Record byte**	**Description**
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=45	3	Object Type = Colour Palette
Options	1	Bitmask	1	0	4	Reserved, sent as zero
Number of ARGB values to follow		Integer	2	0—256	5—6	Indicates the number of Colour Palette entries to follow.
Repeat: {B}		Integer	1	0—255	7 + (No. object*4)...	Blue value
{G}		Integer	1	0—255	8 + (No. object*4)...	Green value
{R}		Integer	1	0—255	9 + (No. object*4)...	Red value
{A}		Integer	1	0—255	10 + (No. object*4)...	Alpha value ranging from 0 (transparent) to 255 (opaque)

B.27 Graphic Data object

The Graphic Data object, available in VT version 6 and later, contains the raw data for a graphic presentation. See Table B.74. This object contains its full colour palette within the object itself, therefore, unlike other objects, this one is not affected by either the Colour Map object or the Colour Palette object.

Allowed commands:

— none.

Table B.74 — Graphic Data attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=46	3	Object Type = Graphic Data
Format	[1]	Integer	1	0	4	Graphic type: 0 = PNG, restricted to 32bit RGBA maximum[15]
Number of bytes in raw data		Integer	4	0 to 2^32—1	5—8	Number of bytes in the raw data.
Repeat: {raw data}		Integer	1	0—255	9...	Raw bytes of graphic data. Bytes shall be interpreted according to the format.

B.28 Scaled Graphic object

The Scaled Graphic object, available in VT version 6 and later, displays a scaled representation of a referenced graphic object. The VT shall scale the raw data from the referenced graphic object from the actual width and height to the target width and height. See Table B.75 and Table B.76. Therefore the target width attribute in the Picture Graphic object shall be ignored when referenced from the Scaled Graphic object.

Allowed commands:

— Change Attribute command;

— Change Numeric Value command (applies to Value attribute);

— Get Attribute Value message.

Table B.75 — Scaled Graphic events

Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Redraw this object.	—
On Change Attribute	Change Attribute command	Redraw this object, refresh parent mask.	Change Attribute response
On Change Value	Change Numeric Value command	If object is displayed, redraw object with new value. Refresh parent object.	Change Numeric Value response

Table B.76 — Scaled Graphic attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=48	3	Object Type = Scaled Graphic
Width	1	Integer	2	0—65535	4—5	Target width in pixels of the graphic.
Height	2	Integer	2	0—65535	6—7	Target height in pixels of the graphic.

Table B.76 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
ScaleType	3	Integer	1	0—127	8	Bits 0 — 2: Scaling Value 0 = Not scaled, use Width and Height as defined in the raw data 1 = Scale to Width maintaining aspect ratio 2 = Scale to Height maintaining aspect ratio 3 = Scale to Width and Height (can distort the image) 4 = Scale to fit the Width and Height maintaining aspect ratio, displaying the largest graphic possible 5—7 = Reserved Field justification. Indicates how the graphic is positioned within the field defined by width and height. Bits 3 — 4: Horizontal Justification Value 0 = Position Left 1 = Position Middle 2 = Position Right 3 = Reserved Bits 5 — 6: Vertical Justification Value 0 = Position Top 1 = Position Middle 2 = Position Bottom 3 = Reserved Bit 7 = Reserved, set to 0.
Options	4	Bitmask	1	0—1	9	Bit 0: 0 = Normal, 1 = Flashing. Flash style and rate determined by VT design. Bits 1—7: = 0, Reserved, set to 0

Table B.76 *(continued)*

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Value	5	Integer	2	0—65535	10—11	Object identifier of a graphic object, or an Object Pointer object. Graphic objects include: — Graphic Data object; — Picture Graphic object. An Object Pointer object shall point only to one of the above listed graphic objects, another Object Pointer object, or the NULL object. To not display any graphic, set the Value attribute to the NULL object id.
Number of macros to follow		Integer	1	0—255	12	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed. VT version 5 and later: A reference to a Macro with 16-bit Object ID shall count as 2 macro references within the context of this attribute.
Repeat: {Event ID}		Integer	1	0—255	13…	(List these after all objects have been listed.) 8-bit Macro Object ID reference: Event ID of event type that causes this Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Event ID of event type that causes this Macro to execute or 0xFF (see 4.6.22.3).
{Macro ID}		Integer	1	0—255	15…	8-bit Macro Object ID reference: Macro ID of the Macro to execute. 16-bit Macro Object ID reference (only for VT version 5 and later): Low byte or high byte of Macro ID of the Macro to execute (see 4.6.22.3).

B.29 Working Set Special Controls object

The Working Set Special Controls object, available in VT version 6 and later, can be used to define the initial Colour Map object and/or the initial Colour Palette object to use for the Working Set, and to define a list of language and country code pairs that supercede the list of languages in the Working Set object. The Working Set Special Controls object is defined to be extensible with the “number of bytes to follow” attribute. This permits forward and backward compatibility by enabling the parsing method to find the next object in the pool, even if it needs to skip over unrecognized capability.

The object pool can contain zero or one Working Set Special Controls object. If this object, or one or more of its attributes (AID 2 and higher) do not exist, as determined by the Number of Bytes to follow attribute, the VT shall use the equivalent to the NULL values for the corresponding attribute(s) (see Table B.78).

The Colour Map object and Colour Palette object references, if not NULL, are activated prior to the first rendering of the Object Pool.

If the Working Set issues a Select Colour Map or Palette command, or if the Working Set issues a Change Attribute command affecting this object, the attributes in this object shall be updated. If the attributes are modified, the VT can need to redraw all or part of the display.

When one or more language and country code pairs are provided in this object, the list of languages in the Working Set object is ignored.

Allowed commands:

— Get Attribute Value message.

Table B.77 — Working Set Special Controls events

Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Affected objects are redrawn.	—

Table B.78 — Working Set Special Controls attributes and record format

Attribute Name	AID	Type	Size (bytes)	Range or Value	Record byte	Description
Object ID		Integer	2	0—65534	1—2	Object identifier. Shall be unique within the object pool.
Type	[0]	Integer	1	=47	3	Object Type = Working Set Special Controls
Number of Bytes to follow	[1]	Integer	2	5—65535	4—5	Number of bytes to follow in this object permits parsing even if this object is extended for future versions.
Object ID of Colour Map object	[2]	Integer	2	0—65534, 65535	6—7	Object ID of a Colour Map object, or NULL to select no Colour Map object.
Object ID of Colour Palette object	[3]	Integer	2	0—65534, 65535	8—9	Object ID of a Colour Palette object, or NULL to select the VT standard colour palette.
Number of languages pairs to follow		Integer	1	0—255	10	The number of language pairs (Language Code: Country Code) to follow.
Repeat: {Language Code}		String	2		11—12	Two-letter code of a supported language (See ISO 639-1) All combinations of a-z and A-Z shall be accepted to guard against changes to the referenced standard.
{Country Code}		String	2		13—14	Two-letter code of a supported country code which is paired with the language (see ISO 3166-1), or 20_{16}, 20_{16} if not applicable. All combinations of a-z and A-Z shall be accepted to guard against changes to the referenced standard.

Annex C
(normative)

Object transport protocol

C.1 Virtual terminal messages and object transfer

Commands are sent from a Working Set to the VT, and from the VT to the Working Set, using the PGNs given in Annex C. Only Working Set Masters (not Members) are allowed to send any of the commands in Annex C. The originating master shall wait for a response before sending another command from Annex C.

Two PGNs are reserved for the VT message protocol, as follows.

— **VT to ECU**

Transmission repetition rate:	As required
Data length:	Variable
Data page field:	0
PDU format field:	230
PDU specific field:	Destination address
Default priority:	5
Parameter group number:	58880 ($00E600_{16}$)

— **ECU to VT**	As required
Transmission repetition rate:	Variable
Data length:	0
Data page field:	231
PDU format field:	Destination address
PDU specific field:	5
Default priority:	59136 ($00E700_{16}$)
Parameter group number:	

The "Default priority" of 5 follows the recommendation given in ISO 11783-3. In VT version 5 and prior, the documented default priority was 7.

Before a Working Set Master builds an object pool in a VT, it can obtain information about the VT's capabilities by using the Get Technical Data messages. Annex D defines messages using the above PGNs to obtain information about the VT's characteristics and thus allow each Working Set Master to configure its object pool to meet the VT's capabilities.

All VT to ECU and ECU to VT messages where the computed data length is less than 8 bytes shall be padded to 8 bytes with FF_{16}.

The VT to ECU and the ECU to VT PGN's are Group Function messages. If a CF receives one of those PGN's but with an unknown or reserved command/parameter defined in the first byte it shall respond using the VT version 5 message Unsupported VT Function message (see G.4) or VT Unsupported VT Function message (see G.5). Prior to VT version 5 the behaviour was undefined.

C.2 Building object pools

C.2.1 General

This clause specifies the transfer of the object pool via an ISO 11783 network. The PGNs listed above are used to transfer the object pool to the VT utilizing single packets (not recommended), the transport and extended transport protocol specified in ISO 11783-3. Destination specific messages shall be used and Connection Management shall be implemented.

The object pool is considered as one large block of data with each object and its attributes making up a single variable length record, as shown in Figure C.1. If the total size of this transfer exceeds the 1 785 byte limit of normal transport protocol, the extended transport protocol specified in ISO 11783-3 shall be used. The VT design shall be able to support all the transport protocol functions.

The VT shall receive, parse and store the received objects. If, during parsing, the VT encounters a new object with an ID matching a previously parsed object, the new object will be considered a replacement for the old object. The VT designer determines the method of storage. The format of the object records was detailed earlier.

Data items and attributes of size greater than 1 byte shall always be transmitted in little endian order (least significant byte first).

Object No. 1	Object ID	Type	Attributes and data
Object No. 2	Object ID	Type	Attributes and data
Object No. 3	Object ID	Type	Attributes and data

Figure C.1 — Object pool variable length record format

The Working Set Master shall transfer a "clean" object pool, adjusted accordingly for the VT hardware connected and also adapted to the VT version being reported. Sending an invalid pool with objects, object references, or macros that do not parse or display properly (e.g. invalid colours) and then altering those objects later with change commands is not permitted, since this can cause parsing errors and error displays at the VT and could cause the VT to ignore those objects or macros with errors or to delete the object pool from volatile storage and suspend the Working Set.

The VT shall identify invalid pools and shall notify the operator about the issue within the object pool. This notification can be deferred to a later time (e.g. when activating a Data Mask containing an invalid reference).

Error Codes in the End of Object Pool response or errors in the VT Change Active Mask message indicate that the VT has identified an invalid object pool.

Even if the VT decides to continue with an invalid object pool, the Working Set Master can decide to go to a safe state when it receives the error indication from the VT.

C.2.2 Object pool transfer procedure

The following sequence shall be used to transfer an object pool.

a) The Working Set Master shall determine if the VT has available memory by transmitting a Get Memory message (see D.2). The VT shall acknowledge this message with a Get Memory response (see D.3). VTs which are designed to do so can use this request to allocate memory for the pool. The Working Set Master shall check the error codes returned.

   1) If no error is reported, the Working Set Master can proceed.

2) If memory is not available, the Working Set Master should repeat this step with a smaller value (e.g. the size of a reduced functionality object pool).

b) The Working Set Master uses single packet (not recommended), transport protocol, or extended transport protocol (specified in ISO 11783-3), or combinations of these, to move the object pool to the VT using the Object pool transfer message (see C.2.3). Normal handshaking, error checking and re-transmission, in accordance with ISO 11783-3, shall be implemented. Working Set designers should recognize that the VT can send CTS with number of packets set to zero (0) while other object pools are being loaded and that this could continue for a significant amount of time.

   The following rules govern the transfer of the object pool:

   1) The Working Set Master can send several single packet, TP or ETP sessions or a combination of any of these to transfer the entire pool. This can be required depending on the size of buffers designed into the Working Set Master. Any number of sessions can be sent before the End of Object Pool message is sent. Multiple TP and/or ETP sessions can also be required if scaling or pool adjustments or both have to be made before sending the object pool to the VT.

   2) Object records in each session shall be complete and shall not be "split" between sessions of TP or ETP. Single packet transfers shall contain a complete object.

   3) Transfer sessions containing no object records are not permitted.

c) The Working Set Master shall transmit an End of Object Pool message (see C.2.4) to the VT to indicate that the object pool is now complete and ready for use.

   1) When the VT receives the End of Object Pool message, it shall set the "parsing" bit in the VT Status message to 1 until it has finished parsing the object pool and sends the End of Object Pool response.

   2) The Working Set Master shall wait for the End of Object Pool response message until three consecutive VT Status messages have been received where the "parsing" bit is set to 0. Three messages are waited for to avoid race conditions created by a VT Status message that can already be in a transmit queue, which does not correctly identify the parsing state.

   3) If the End of Object Pool response is not received by the Working Set Master, then it shall assume that the End of Object Pool message was not received by the VT. Under these conditions the Working Set Master can retry the End of Object Pool message up to three times before it assumes an unexpected shutdown of the VT after which the Working Set Master shall obey the requirements of 4.6.9 (Connection management) (Updating pools at runtime).

   4) If the End of Pool response indicates the VT removed the pool from memory, the Working Set Master can retry the process (e.g. Working Set Master selects a reduced functionality object pool after receiving an out of memory response).

C.2.3 Object pool transfer message

The following message is sent by a Working Set Master to transfer part of an object pool to the VT.

Transmission repetition rate:	As required
Data length:	Variable
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 17_{10}		
	Bits	7—4 0001	Command	Object Pool Transfer
	Bits	3—0 0001	Parameter	Object Pool Transfer
Bytes	2—n		Object pool records (see Figure C.1)	

NOTE Since there is no response to this message, it is recommended to not send single objects that fit within a single packet.

C.2.4 End of Object Pool message

The following message is sent by a Working Set Master to indicate that the object pool is complete and ready for use. It is sent after the initial object pool definition and also after any object is redefined or added to the pool during operation.

Transmission repetition rate:	Upon completion of object pool transfer
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 18_{10}			
	Bits	7—4	0001	Command	Object Pool Transfer
	Bits	3—0	0010	Parameter	Object Pool Ready
Bytes	2—8			Reserved, set to FF_{16}	

C.2.5 End of Object Pool response

This message is sent by the VT to a Working Set Master to acknowledge the End of Object Pool message. When the VT replies with an error of any type, the VT should delete the object pool from volatile memory storage and inform the operator by an alarm type method of the suspension of the Working Set and indicate the reason for the deletion. On reception of this message, the responsible ECU(s) should enter a fail-safe operation mode providing a safe shutdown procedure of the whole device.

NOTE A VT can take a long time to parse a pool.

The End of Object Pool response shall be delayed until this activity completes. The VT Status message shall reflect the current state of the VT (is busy parsing a pool).

Transmission repetition rate:	In response to End of Object Pool message
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 18_{10}			
	Bits	7—4	0001	Command	Object Pool Transfer
	Bits	3—0	0010	Parameter	Object Pool Ready
Byte	2			Error Codes (0 = no errors) Bit 0 = 1 = There are errors in the Object Pool, refer to Bytes 3 to 8 for additional error information Bit 1 = 1 = VT ran out of memory during transfer Bit 2, 3 = Reserved, set to 0 Bit 4 = 1 = any other error Bit 5—7 = Reserved, set to 0	
Bytes	3, 4			Parent Object ID of faulty object, set to NULL Object ID if there are no object pool errors	

Bytes	5, 6	Object ID of faulty object, set to NULL Object ID if there are no object pool errors
Byte	7	Object Pool Error Codes (0 = no errors) Bit 0 = 1 = method or attribute not supported by the VT Bit 1 = 1 = unknown object reference (missing object) Bit 2 = 1 = any other error Bit 3 = 1 = object pool was deleted from volatile memory Bit 4—7 = Reserved, set to 0
Byte	8	Reserved, set to FF_{16}

C.2.6 Updating pools at runtime

If the Working Set needs to modify or add one or more objects, then the Working Set can update its pool at runtime. For example, if the Working Set needs to change the size of an object (e.g. increase the length of a string object) it can send the replacement object with the revised record format (see 4.6.10.3). This is accomplished by using the same messages and procedures used to upload the pool at initialization as follows.

a) The Working Set Master shall determine if the VT has available memory by transmitting a Get Memory message (see D.2). The Memory Required parameter shall be based on the size of this update to the pool, and not based on the size of the original pool plus the update. The VT acknowledges this message with a Get Memory response (see D.3). The Working Set Master shall check the error codes returned.

   1) If no error is reported, the Working Set Master can proceed.

   2) If memory is not available, the Working Set Master determines the recovery (e.g. delete the pool and send a reduced functionality object pool).

b) The Working Set Master uses single packet transfer (not recommended), extended transport protocol, or transport protocol, to move the object or objects to the VT. Normal handshaking, error checking and retransmission, in accordance with ISO 11783-3, shall be implemented [see C.2.2.b)].

c) The Working Set Master shall transmit an End of Object Pool message (see C.2.4) to the VT to indicate that the update is now complete and ready for use. The process continues as in C.2.2 Object pool transfer procedure step 3.

Only those objects that need to be changed should be transmitted during the update; all other objects will remain in VT memory following the update.

In the case of errors in the update to the object pool, the VT indicates the errors with the End of Object Pool response. The VT deletes the entire object pool from volatile memory, (including the object pool as it existed prior to the object pool update) and informs the operator by an alarm type method of the suspension of the Working Set and indicates the reason for the deletion. On reception of this message, the responsible ECU(s) shall behave as described (see C.2.5) when an End of Object Pool response is received indicating errors in the object pool.

The VT shall handle commands and macros from a Working Set even while that Working Set is updating its object pool.

EXAMPLE The operator presses a Soft Key while the pool is being updated by the Working Set. The VT immediately executes the macros triggered and commands sent from that Working Set without waiting for the completion of the pool update.

The VT shall keep the new/updated objects separate from the original pool, until reception of the End of Object Pool message. The objects shall be merged into the original pool before the VT sends the End of Object Pool response. Working Set designers shall be aware that behaviour is unpredictable for commands acting on the new/updated objects if these commands are sent after the End of Object Pool message and before reception of End of Object Pool response. Such commands will be applied to either the original pool or the updated pool. Changes to objects shall take effect immediately.

It is recommended that Working Sets do not transmit commands which act upon these new objects until the End of Object Pool response is received from the VT.

Changes to objects should utilize Annex F commands when possible rather than performing a pool update due to the processing time of the pool update process (e.g. Change Size command performs faster than reloading an object with a new size).

Annex D
(normative)

Technical data messages

D.1 General

The technical data messages are used to request the characteristics of the VT. They consist of the request for data by a CF and the response by the VT. The following messages are not allowed in macros. Working Set masters, members, or any other CF can send any command in this annex. The parameters in the response messages can depend on the Version Number of the VT and shall not be adapted according to the Working Set's Version Number[5].

VT hardware dependent characteristics that are dependent on the VT version number are identified in 4.1.1.

D.2 Get Memory message

The Get Memory message allows the CF to determine if the VT is out of memory and/or to determine the VT version.

In version 3 and prior VTs, the "Memory Required" parameter represents the number of bytes in the object pool (see C.2.1) to be transferred. In version 4 and later VTs supporting the Graphics Context object, the "memory required" parameter is the sum of the number of bytes in the object pool to be transferred and the estimated storage of all Graphics Context objects that are defined in the object pool to be transferred. The Working Set can send less than the amount communicated in the Memory Required parameter.

The storage space required for a single Object Pool object (OPO) can be determined from the definition of the specific object.

The storage space required for a single Graphics Context object (GCO) shall be estimated as follows:

Size of a GCO =

$$\mathrm{RoundUp}\left(\frac{\mathrm{GCO.width}}{\mathrm{VT.PixelsPerByte}}\right)\mathrm{GCO.height}$$

where

RoundUp	is a function that will round the value up to the nearest integer, width and height are measured in pixels;
VT.PixelsPerByte	is based on the VT capabilities and is defined as follows:

VT Capability	Graphic Type	VT.PixelsPerByte
Monochrome	0	8
16 colour	1	2
256 colour	2	1

5) The Working Set's Version Number might not yet be known by the VT.

The Memory Required parameter can then be estimated as follows:

Memory Required =

$$\left\{\sum_{i=1}^{\text{Number of OPO}} \text{size of }\left(\text{OPO}\left[i\right]\right)\right\}+\left\{\sum_{j=1}^{\text{Number of GCO}} \text{size of }\left(\text{GCO}\left[j\right]\right)\right\}$$

where the second parenthesized term is only used for VT version 4 and later.

For more details on the use of the Memory Required parameter, see C.2.2 for Object pool transfer procedure and C.2.6 for Updating pools at runtime.

The CF should send the Get Memory message with the Memory Required set to zero in order to receive the response indicating the VT version number. This information can be used to calculate the actual Memory Required parameter.

If the VT design allocates memory based on this message, it should not allocate any storage when the Memory Required is zero, or in the case where a CF has sent this message but has not been identified by the VT as a Working Set Master (see 4.6.6.2).

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 192_{10}			
	Bits	7—4	1100	Command	Get Technical Data
	Bits	3—0	0000	Parameter	Get Memory
Bytes	2			Reserved, set to FF_{16}	
Bytes	3—6			Memory Required	
Bytes	7, 8			Reserved, set to FF_{16}	

D.3 Get Memory response

The Version Number, reported in this message, indicates the capabilities of the VT. It is the Working Set's responsibility to ensure that it does not use features not supported by the VT. The versions are backwards compatible, and therefore a Working Set designed for version N does not have to implement special handling for VTs with versions higher than N.

The list of versions below indicates the document revision, when the version was introduced, but new designs should use the latest revision of the document, even when implementing to an older version.

If the VT responds with status code one (1), the Working Set Master shall not transmit its object pool.

Transmission repetition rate:	In Response to Get Memory message
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 192_{10}		
	Bits	7—4 1100	Command	Get Technical Data
	Bits	3—0 0000	Parameter	Get Memory
Byte	2	Version Number	The version of ISO 11783-6 that this VT meets 0 = Hannover Agritechnica 2001 limited feature set 1 = Introduced in ISO11783-6:2002 (E), FDIS Version 2 = Introduced in ISO11783-6:2004 (E), First Edition 3 = Introduced in ISO11783-6:2010 (E), Second Edition 4 = Introduced in ISO11783-6:2010 (E), Second Edition 5 = Introduced in ISO11783-6:2014(E), Third Edition 6 = Introduced in ISO11783-6:2018 (E), Fourth Edition	
Byte	3		Status 0 = There can be enough memory[a]. 1 = There is not enough memory available. Do not transmit Object Pool.	
Bytes	4—8		Reserved, set to FF_{16}	

[a] Because there is overhead associated with object storage it is impossible to predict whether there is enough memory available without having prior knowledge of the exact content of the object pool.

D.4 Get Number of Soft Keys message

The Get Number of Soft Keys message is used by the CF to request the available divisions of the X and Y axes for Soft Key descriptors, the available virtual Soft Keys and the number of physical Soft Keys. VT Version 4 and later provides the number of physical Soft Keys that are used by the VT for navigation among the virtual Soft Keys.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 194_{10}		
	Bits	7—4 1100	Command	Get Technical Data
	Bits	3—0 0010	Parameter	Get Number of Soft Keys
Bytes	2—8		Reserved, set to FF_{16}	

D.5 Get Number of Soft Keys response

Transmission repetition rate:	In response to Get Number of Soft Keys message
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 194_{10}		
	Bits	7—4 1100	Command	Get Technical Data
	Bits	3—0 0010	Parameter	Get Number of Soft Keys Response
Byte	2	Navigation Soft Keys	Version 3 and Prior: Reserved, set to FF16 Version 4 and Later: The number of Physical Soft Keys that are used by the VT for navigation among the Virtual Soft Keys.	
Bytes	3—4		Reserved, set to FF_{16}	
Byte	5	X Dots	Number of pixels on the x-axis for a Soft Key descriptor	
Byte	6	Y Dots	Number of pixels on the y-axis for a Soft Key descriptor	
Byte	7	Virtual Soft Keys	Number of possible virtual Soft Keys in a Soft Key Mask Version 3 and Prior: 6 to 64 (inclusive) Version 4 and Later: 64	
Byte	8	Physical Soft Keys	Number of Physical Soft Keys	

D.6 Get Text Font Data message

The Get Text Font Data message allows the CF to request the characteristics of fonts, type sizes, type attributes and colour capabilities.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 195_{10}		
	Bits	7—4 1100	Command	Get Technical Data
	Bits	3—0 0011	Parameter	Get Technical Data
Bytes	2—8		Reserved, set to FF_{16}	

D.7 Get Text Font Data response

Transmission repetition rate:	In response to Get Text Font Data message
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 195_{10}			
	Bits	7—4	1100	Command	Get Technical Data
	Bits	3—0	0011	Parameter	GetText Font Data Response
Bytes	2—5			Reserved, set to FF_{16}	
Byte	6	Small font sizes			(Values are width × height)
				0000 0000	Font 6 × 8 (Default)
				0000 0001	Font 8 × 8
				0000 0010	Font 8 × 12
				0000 0100	Font 12 × 16
				0000 1000	Font 16 × 16
				0001 0000	Font 16 × 24
				0010 0000	Font 24 × 32
				0100 0000	Font 32 × 32
				1000 0000	Reserved
Byte	7	Large font sizes			
				0000 0001	Font 32 × 48
				0000 0010	Font 48 × 64
				0000 0100	Font 64 × 64
				0000 1000	Font 64 × 96
				0001 0000	Font 96 × 128
				0010 0000	Font 128 × 128
				0100 0000	Font 128 × 192
				1000 0000	Reserved
Byte	8	Font styles			
				0000 0000	Normal text (Default)
				0000 0001	Bold text
				0000 0010	Crossed out text
				0000 0100	Underlined text
				0000 1000	Italics text
				0001 0000	Inverted text
				0010 0000	Flash between inverted and styles set by bits 0—3
				0100 0000	Flash both the background and the foreground between Hidden and styles set by bits 0—4
				1000 0000	Proportional font rendering[a]

[a] VT version 4 and later, and as a bitmask, this bit can be sent to a VT version 3 and prior Working Set.

D.8 Get Hardware message

The CF sends the Get Hardware message to request the hardware design of the VT.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 199_{10}			
	Bits	7—4	1100	Command	Get Technical Data
	Bits	3—0	0111	Parameter	Get Hardware
Bytes	2—8			Reserved, set to FF_{16}	

D.9 Get Hardware response

Transmission repetition rate:	In response to Get Hardware message
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 199_{10}			
	Bits	7—4	1100	Command	Get Technical Data
	Bits	3—0	0111	Parameter	Get Hardware Response
Byte	2	Boot time[a]		Maximum number of seconds from a VT power startup or reset cycle to the transmission of the first "VT Status message" (see 4.6.6.3, and 4.6.9). Set to FF_{16} when this information is not available.	
Byte	3	Graphic Type		Supported graphic modes 0 = Monochrome (VT supports colour codes 0 and 1 and monochrome Picture Graphic objects only). 1 = 16 Colour (VT supports colour codes 0 through 15 and monochrome and 16 colour Picture Graphic objects). 2 = 256 Colour (VT supports colour codes 0 through 255 and all formats of Picture Graphic objects).	

Byte	4	Hardware	Supported hardware features Bit 0 = 1 = VT has a touch screen and supports Pointing Event message. Bit 1 = 1 = VT has a pointing device and supports Pointing Event message. Bit 2 = 1 = VT has multiple frequency audio output. Bit 3 = 1 = VT has adjustable volume audio output. Bit 4 = 1 = VT supports simultaneous activations of all combinations of Physical Soft Keys (see 4.6.18 Soft Key and Button activation)[b]. Bit 5 = 1 = VT supports simultaneous activations of all combinations of Buttons (see 4.6.18 Soft Key and Button activation)[b]. Bit 6 = 1 = VT reports drag operation via Pointing Event message (Bit 0 or Bit 1 shall be set to 1)[b]. Bit 7 = 1 = VT supports intermediate coordinates during a drag operation (Bit 6 shall be set to a 1)[b].
Bytes	5, 6	X-Pixels	Number of divisions on the horizontal axis (X dots) (16 bit unsigned integer) in the Data Mask Area
Bytes	7, 8	Y-Pixels	Number of divisions on the vertical axis (Y dots) (16 bit unsigned integer) in the Data Mask Area. Since the Data Mask is square, this value is always the same as the X Value.

[a] Introduced for VT version 4 and later. Allowed in VT versions prior to 4.

[b] These bits exist in VT version 4 and later.

D.10 Get Supported Widechars message

This message only applies to version 4 and later VTs.

The Get Supported Widechars message is used by the CF to determine the WideChars supported by the VT.

The message only requests characters from a single code plane. If the CF requires information about multiple code planes multiple messages shall be sent.

The request contains First WideChar and Last WideChar as a range, where First WideChar ≤ Last WideChar.

The CF can reduce the size of the response frame by sending multiple requests with small inquiry ranges instead of one request with a large inquiry range.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 193_{10}			
	Bits	7—4	1100	Command	Get Technical Data
	Bits	3—0	0001	Parameter	Get Supported WideChars
Byte	2			Code plane	0 => characters 00000_{16} — $0FFFF_{16}$ 1 => characters 10000_{16} — $1FFFF_{16}$ ... 16 => characters $10\ 0000_{16}$ — $10\ FFFF_{16}$
Bytes	3, 4			First WideChar in inquiry range	
Bytes	5, 6			Last WideChar in inquiry range	
Bytes	7, 8			Reserved, set to FF_{16}	

D.11 Get Supported WideChars response

This message only applies to version 4 and later.

Transmission repetition rate:	In response to Get Supported Widechars message
Data length:	Variable
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 193_{10}		
	Bits	7—4 1100	Command	Get Technical Data
	Bits	3—0 0001	Parameter	Get Supported WideChars response
Byte	2		Code plane	0 => characters 0000_{16} — $0FFFF_{16}$ 1 => characters 10000_{16} — $1FFFF_{16}$ ... 16 => characters 10 0000_{16} — 10 $FFFF_{16}$
Bytes	3, 4		First WideChar in inquiry range	
Bytes	5, 6		Last WideChar in inquiry range	
Byte	7		Error Codes (0 = no errors) Bit 0 = 1 = Too many ranges (more than 255 sub-ranges in the requested range) Bit 1 = 1 = Error in Code plane Bits 2-3 = Reserved, set to 0 Bit 4 = 1 = any other error Bits 5—7 = Reserved, set to 0	
Byte	8	Number of ranges	Indicates the number of entries in the WideChar range array. Set to 0 if Error codes is not equal to 0.	
Bytes	9—n	WideChar range array	Each entry in the array consists of two WideChars: first WideChar, last WideChar.	

The CF does not have to request this message because the VT displays WideStrings even if they contain unsupported characters (see 4.6.19.7). The VT includes the characters from the WideChar minimum character set when responding to a Code Plane 0 request (see Table K.8).

EXAMPLE Response from a VT supporting only the WideChar Minimum Character Set. The CF has requested information about characters 0000_{16} to $3FFF_{16}$ in code plane 0.

$C1_{16}$,	; Command
00_{16},	; Code plane 0
00_{16}, 00_{16}, FF_{16}, $3F_{16}$,	; Inquiry range (0000_{16}—$03FFF_{16}$)
00_{16},	; Error Codes
$0F_{16}$,	; 15 ranges
20_{16}, 00_{16}, $7E_{16}$, 00_{16},	; Range 1. Character 0020_{16}—$007E_{16}$
$A0_{16}$, 00_{16}, $7E_{16}$, 01_{16},	; Range 2. Character $00A0_{16}$—$017E_{16}$
$C6_{16}$, 02_{16}, $C7_{16}$, 02_{16},	; Range 3. Character $02C6_{16}$—$02C7_{16}$
$C9_{16}$, 02_{16}, $C9_{16}$, 02_{16},	; Range 4. Character $02C9_{16}$—$02C9_{16}$
$D8_{16}$, 02_{16}, DD_{16}, 02_{16},	; Range 5. Character $02D8_{16}$—$02DD_{16}$
$7E_{16}$, 03_{16}, $7E_{16}$, 03_{16},	; Range 6. Character $037E_{16}$—$037E_{16}$
84_{16}, 03_{16}, $8A_{16}$, 03_{16},	; Range 7. Character 0384_{16}—$038A_{16}$
$8C_{16}$, 03_{16}, $8C_{16}$, 03_{16},	; Range 8. Character $038C_{16}$—$038C_{16}$
$8E_{16}$, 03_{16}, $A1_{16}$, 03_{16},	; Range 9. Character $038E_{16}$—$03A1_{16}$
$A3_{16}$, 03_{16}, CE_{16}, 03_{16},	; Range 10. Character $03A3_{16}$—$03CE_{16}$
01_{16}, 04_{16}, $0C_{16}$, 04_{16},	; Range 11. Character 0401_{16}—$040C_{16}$
$0E_{16}$, 04_{16}, $4F_{16}$, 04_{16},	; Range 12. Character $040E_{16}$—$044F_{16}$
51_{16}, 04_{16}, $5C_{16}$, 04_{16},	; Range 13. Character 0451_{16}—$045C_{16}$
$5E_{16}$, 04_{16}, $5F_{16}$, 04_{16},	; Range 14. Character $045E_{16}$—$045F_{16}$
AC_{16}, 20_{16}, AC_{16}, 20_{16},	; Range 15. Character $20AC_{16}$—$20AC_{16}$

D.12 Get Window Mask Data message

This message applies to version 4 and later VTs that support the Window Mask object (as reported by the Get Supported Objects response).

The CF sends the Get Window Mask Data message to request the background colour of User-Layout Data Mask and the background colour of the Key Cells on a User-Layout Soft Key Mask.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 196_{10}			
	Bits	7—4	1100	Command	Get Technical Data
	Bits	3—0	0100	Parameter	Get Window Mask Data
Bytes	2—8			Reserved, set to FF_{16}	

D.13 Get Window Mask Data response

This message only applies to version 4 and later VTs that support the Window Mask object (as reported by the Get Supported Objects response).

Transmission repetition rate:	In response to Get Window Mask Data message
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 196_{10}			
	Bits	7—4	1100	Command	Get Technical Data
	Bits	3—0	0100	Parameter	Get Window Mask Data
Byte	2			Background colour of VT's User-Layout Data Masks.	
Byte	3			Background colour of VT's Key Cells when on a User-Layout Soft Key Mask.	
Bytes	4—8			Reserved, set to FF_{16}	

D.14 Get Supported Objects message

This command is used by the CF to get the list of all object types supported by the VT (see 4.6.28).

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 197_{10}			
	Bits	7—4	1100	Command	Get Technical Data
	Bits	3—0	0101	Parameter	Get Supported Objects
Bytes	2—8			Reserved, set to FF_{16}	

D.15 Get Supported Objects response

This message is available in VT version 4 and later.

The VT uses this message to respond to a Get Supported Objects message (see 4.6.28).

The VT shall return a list of all supported object types including (see Table A.1):

— all object types that are mandatory for the VT to support;

— all optional object types that the VT supports;

— proprietary object types that the VT supports, which can vary depending on the connected WS, or might not be listed at all.

The VT and WS can decide by another means which proprietary objects can be supported.

In either case whether the VT lists the proprietary objects in this message or not, the VT can still decide to reject the object pool from the WS because it included proprietary objects.

The WS can also decide by another means that it does not want to include the proprietary objects listed by the VT in its object pool.

VTs shall not list Auxiliary Input Type 1 and Auxiliary Function Type 1 objects in the list of supported objects.

Transmission repetition rate:	In response to Get Supported Objects message
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 197_{10}			
	Bits	7—4	1100	Command	Get Technical Data
	Bits	3—0	0101	Parameter	Get Supported Objects Response
Byte	2			Number of bytes to follow. NOTE For future compatibility, this is NOT necessarily the number of Object Types.	
Bytes	3—n			Numerically ascending sorted list of all Object Types supported by the VT. Each Object Type is an unsigned integer occupying a single byte. If the special value FF_{16} is found by the WS during parsing, it indicates the end of the list, and all following bytes should be ignored.	

D.16 Screen Capture command

This command, available in VT version 6 and later, requests a lossless screen (image) capture from the VT. When this command is accepted by a VT, it will capture the current screen image and communicate the results of the capture via the Screen Capture response.

The Screen Capture mechanism is intended primarily for test system usage. Therefore, the VT designer can choose to disable this mechanism by default and require specific proprietary interactions to enable it. An error code in the response is used to indicate the disabled state.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 198_{10}			
	Bits	7—4	1100	Command	Command
	Bits	3—0	0110	Parameter	Screen Capture command
Byte	2			Item Requested 0 = Screen Image 1 — 239 = Reserved 240 — 255 = Manufacturer Proprietary	
Byte	3			Path 0 = Reserved 1 = VT accessible storage/removable media 2 — 239 = Reserved 240 — 255 = Manufacturer Proprietary	
Bytes	4—8			Reserved, sent as FF_{16}	

D.17 Screen Capture response

This message is used to respond to a Screen Capture command.

If the VT supports delivering the lossless screen capture to removable media (e.g. SD card, USB memory device) in a common file format (e.g. PNG, BMP), the response includes an unsigned 16 bit value as the image identification number. The VT shall encode this number in the name of the stored file (e.g. IMG00001.PNG, IMG00002.PNG). Additionally, the identification number shall increment for each captured image. Based on the VT design and/or the initial contents of the file system, the number might not start at one.

Transmission repetition rate:	In response to a Screen Capture command
Data length:	Variable
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 198_{10}			
	Bits	7—4	1100	Command	Command
	Bits	3—0	0110	Parameter	Screen Capture response
Byte	2			Item Requested Item requested from the Screen Capture command	
Byte	3			Path Path requested from the Screen Capture command	

Byte	4	Error Codes (0 = no errors) Bit 0 = 1 = Screen Capture is not enabled Bit 1 = 1 = VT Transfer buffer is busy Bit 2 = 1 = Item — unsupported request Bit 3 = 1 = Path — unsupported request Bit 4 = 1 = Removable media path is full or unavailable Bit 5 = 1 = Any other error

If Item and Path are not Manufacturer proprietary:

If Error Codes = = 0 and Path = = 1:

Bytes	5, 6	Image Identification number (0 — 65535_{10})
Bytes	7, 8	Reserved, sent as FF_{16}

If Error Codes ! = 0:

Bytes	5—8	Reserved, sent as FF_{16}

If Item is Manufacturer Proprietary or Path is Manufacturer proprietary:

Bytes	5—n	Manufacturer proprietary.

D.18 Identify VT message

Upon receipt of this message and only if no Alarm Mask is currently active, the VT shall display, for a period of 3 s, the VT Number (see Clause 3). This message is intended to be sent Destination-Global, however it can be sent Destination-Specific.

The presentation of the VT Number is considered VT proprietary and the VT designer can choose to present other information indicating the purpose of the VT Number (see 4.6.25).

The VT Number shall be in the range of 1 to 32, corresponding to Function Instances 0 to 31. VTs can then be referenced as VT Number 1, VT Number 2, etc.

VT Number = VT Function Instance + 1.

NOTE 1 The offset of 1 is in support of operators, which might not be familiar with a zero-based numbering system.

NOTE 2 This message is available in VT version 4 and later.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Global (Destination-Specific)
Allowed in a Macro:	No

Byte	1	VT function = 187_{10}		
	Bits	7—4 1011	Command	Command
	Bits	3—0 1011	Parameter	Identify VT
Bytes	2—8		Reserved, set to FF_{16}	

D.19 Identify VT response

The VT uses this message to respond to the Identify VT message if it was received Destination-Specific.

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	In response to Identify VT message
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 187_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	1011	Parameter	Identify VT
Bytes	2—8			Reserved, set to FF_{16}	

Annex E
(normative)

Non-volatile memory operations commands

E.1 General

E.1.1 Introduction

The VT provides functions to store and to restore a complete Working Set-specific object pool. When connecting to the VT, the Working Set can send a message to get its object pool copied from non-volatile storage into volatile storage. The availability and organization of the non-volatile storage area is VT-specific. Storing and restoring an object pool includes all object definitions. There shall be a method inside the VT to assign a stored object pool uniquely to a specific Working Set. If the VT is capable of changing the characteristics in Annex D (e.g. Data Mask size), then there shall be a method inside the VT to either associate a stored pool with a specific set of characteristics or delete any stored pools. Dependant on the VT design, either only a single object pool or an arbitrary number of pools can be managed for each Working Set. Each pool should be identified by a version label.

Version labels can be displayed to an operator and can be used as a file name. As such the Working Set shall apply the following rules. Version labels shall be constructed of visible characters from font type zero (see Table K.2 — ISO 8859-1 (Latin 1) character set). Version labels shall be padded with trailing blanks to the defined version label size. In addition, the following characters shall not be used in a version label string:

\	[5C$_{16}$] Reverse Solidus (Back slash)
"	[22$_{16}$] Quotation mark (Double quote)
'	[27$_{16}$] Apostrophe (Single quote)
`	[60$_{16}$] Grave Accent (Back tic)
/	[2F$_{16}$] Solidus (Forward slash)
:	[3A$_{16}$] Colon
*	[2A$_{16}$] Asterisk
<	[3C$_{16}$] Less-than sign
>	[3E$_{16}$] Greater-than sign
|	[7C$_{16}$] Vertical line
?	[3F$_{16}$] Question mark

In order to maintain different versions of object pools in the non-volatile storage of the VT, the Working Set needs to detect those versions currently stored by the VT. It shall also determine if any of the available versions are suitable for the Working Sets current software version before an object pool is copied into the object buffer of the VT.

The Working Set shall be identified by the entire ISO NAME of the Working Set Master. Each Working Set Master shall only be allowed to manipulate its own versions of object pools and shall not perform any of the commands in this annex on versions of object pools created by other Working Sets. Only Working Set Masters (not Members) are allowed to send any of the commands in this annex.

Because non-volatile operations can take an indefinite amount of time to complete, the response message can be delayed accordingly. Therefore the VT Status message reflects the current state of the VT (i.e. is busy). Only when the VT has completed the non-volatile operations shall it send the response message.

The messages in this annex are not allowed in macros.

E.1.2 Version Management — VT version 4 and prior

Version labels are defined as a seven character 8-bit string. The permissible messages do not include the extended versions of the messages in this annex.

E.1.3 Version Management — VT version 5 and later

Version labels are defined as either a seven character 8-bit string, or a 32 character 8-bit string, and the appropriate messages in this annex shall be used accordingly.

E.2 Get Versions message

The Get Versions message allows the Working Set to query the VT for existing seven-character version labels associated with the requesting Working Set.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 223_{10}			
	Bits	7—4	1101	Command	Non Volatile Memory
	Bits	3—0	1111	Parameter	Get Versions
Bytes	2—8			Reserved, set to FF_{16}	

E.3 Get Versions response

The VT sends all seven-character version labels contained in the non-volatile storage associated with the requesting Working Set. If there is no seven-character version stored, the number of version strings is set to 0 and the remaining bytes in the packet shall be set to FF_{16}. Extended Transport Protocol and Transport Protocol are used when necessary.

Transmission repetition rate:	In response to Get Versions message
Data length:	Variable
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 224_{10}		
	Bits	7—4 1110	Command	Non Volatile Memory
	Bits	3—0 0000	Parameter	Get Versions response
Byte	2		Number of version strings to follow (each is 7 bytes)	
Bytes	3—n	Version labels	7 character version strings, unused bytes filled with spaces. Only 8-bit Strings are allowed.	

E.4 Store Version command

The Store Version command allows a Working Set to store the copy of the actual object pool into the non-volatile storage of the VT. This message can be sent at any time. The copy is stored as the version indicated by version label. If a copy with the same version label already exists in the non-volatile storage area, it is overwritten. All objects are stored as they are (with current attributes, input values etc.). If the version label contains no string (all blanks) the last stored seven-character version in non-volatile storage shall be overwritten; alternatively, if there is no seven-character version stored up to that point, an error shall be indicated by the VT.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 208_{10}		
	Bits	7—4 1101	Command	Non Volatile Memory
	Bits	3—0 0000	Parameter	Store Version
Bytes	2—8	Version label	7 character version string, unused bytes filled with spaces. Only 8-bit Strings are allowed.	

E.5 Store Version response

The VT acknowledges whether the object pool was stored in the non-volatile storage.

Transmission repetition rate:	In response to Store Version command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 208_{10}			
	Bits	7—4	1101	Command	Non Volatile Memory
	Bits	3—0	0000	Parameter	Store Version Response
Bytes	2—5			Reserved, set to FF_{16}	
Byte	6			Error Codes (0 = no errors; successfully stored) Bit 0 = Reserved Bit 1 = 1 = Version label is not correct Bit 2 = 1 = Insufficient memory available Bit 3 = 1 = Any other error	
Bytes	7, 8			Reserved, set to FF_{16}	

E.6 Load Version command

The Load Version command allows a Working Set to load a copy of an object pool from the non-volatile storage of the VT. If an object pool is already loaded it is overwritten. If the message is acknowledged positive by the VT, the Working Set can proceed as if all objects had been transmitted normally. If the version label contains no string (all blanks), the last stored seven-character version in non-volatile storage shall be loaded.

When the VT receives the Load Version command, it shall set the “parsing” bit in the VT Status message to 1 until it has finished parsing the object pool and sends the Load Version Response message.

The Working Set Master shall wait for the Load Version response until three consecutive VT Status messages have been received where the “parsing” bit is set to 0. At that time, if the Load Version response has not been received by the Working Set Master, then it shall assume that the Load Version command was not received by the VT. Three messages are waited for to avoid race conditions created by a VT Status message that can already be in a transmit queue, which does not correctly identify the parsing state.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 209_{10}			
	Bits	7—4	1101	Command	Non Volatile Memory
	Bits	3—0	0001	Parameter	Load Version
Bytes	2—8	Version label		7 character version string, unused bytes shall be filled with spaces. Only 8-bit Strings are allowed.	

E.7 Load Version response

The VT acknowledges whether a copy was loaded from the non-volatile storage.

Transmission repetition rate:	In response to Load Version command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 209_{10}			
	Bits	7—4	1101	Command	Non Volatile Memory
	Bits	3—0	0001	Parameter	Load Version Response
Bytes	2—5			Reserved, set to FF_{16}	
Byte	6			Error Codes (0 = no errors; successfully loaded) Bit 0 = 1 = File system error or pool data corruption[a] Bit 1 = 1 = Version label is not correct or Version label unknown Bit 2 = 1 = Insufficient memory available Bit 3 = 1 = Any other error	
Bytes	7, 8			Reserved, set to FF_{16}	

[a] This bit exists in VT version 4 and later.

E.8 Delete Version command

The Delete Version command allows a Working Set to delete a seven-character version of an object pool in the non-volatile storage of the VT. If a copy of this version is in the volatile memory at the same time it is preserved there — this message affects non-volatile storage only.

If the version label contains no string (all blanks) the last stored seven-character version in non-volatile storage is to be deleted. Repeating this command causes each successive “last stored version” to be deleted.

If the version label contains a single asterisk character (*) padded right with 6 blanks this shall be interpreted by the VT as a wild card and shall cause the deletion of all seven-character version object pools in NVM owned by the requesting Working Set. It does not delete object pools owned by other Working Sets[6)].

NOTE To delete the object pool from volatile memory, see F.44.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 210_{10}			
	Bits	7—4	1101	Command	Non Volatile Memory
	Bits	3—0	0010	Parameter	Delete Version
Bytes	2—8	Version label		7 character version string, unused bytes shall be filled with spaces. Only 8-bit Strings are allowed.	

6) The asterisk wild-card delete-all function is available in VT Version 6 and later.

E.9 Delete Version response

The VT acknowledges whether a version was deleted in the non-volatile storage.

Transmission repetition rate:	In response to Delete Version command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 210_{10}			
	Bits	7—4	1101	Command	Non Volatile Memory
	Bits	3—0	0010	Parameter	Delete Version response
Bytes	2—5			Reserved, set to FF_{16}	
Byte	6			Error Codes (0 = no errors; successfully deleted or wild card delete was commanded) Bit 0 = Reserved Bit 1 = 1 = Version label is not correct or Version label unknown Bit 2 = Reserved Bit 3 = 1 = Any other error	
Bytes	7, 8			Reserved, set to FF_{16}	

E.10 Extended Get Versions message

The Extended Get Versions message allows the Working Set to query the VT for existing extended version labels associated with the requesting Working Set.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 211_{10}			
	Bits	7—4	1101	Command	Non Volatile Memory
	Bits	3—0	0011	Parameter	Extended Get Versions
Bytes	2—8			Reserved, set to FF_{16}	

NOTE This message is available in VT version 5 and later.

E.11 Extended Get Versions response

The VT sends all extended version labels contained in the non-volatile storage associated with the requesting Working Set. If there is no extended version stored, the number of version strings is set to 0 and the remaining bytes in the packet shall be set to FF_{16}. Extended Transport Protocol and Transport Protocol are used when necessary.

Transmission repetition rate:	In response to Extended Get Versions message
Data length:	Variable
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 211_{10}			
	Bits	7—4	1101	Command	Non Volatile Memory
	Bits	3—0	0011	Parameter	Extended Get Versions
Byte	2			Number of version strings to follow (each is 32 bytes)	
Bytes	3—n	Version labels		32 character version strings, unused bytes shall be filled with spaces. Only 8-bit Strings are allowed.	

NOTE This message is available in VT version 5 and later.

E.12 Extended Store Version command

The Extended Store Version command allows a Working Set to store the copy of the actual object pool into the non-volatile storage of the VT. This message can be sent at any time. The copy is stored as the version indicated by an extended version label. If a copy with the same version label already exists in the non-volatile storage area, it is overwritten. All objects are stored as they are (with current attributes, input values etc.). If the version label contains no string (all blanks) the last stored extended version in non-volatile storage shall be overwritten; alternatively, if there is no extended version stored up to that point, an error shall be indicated by the VT.

Transmission repetition rate:	On request
Data length:	33 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 212_{10}			
	Bits	7—4	1101	Command	Non Volatile Memory
	Bits	3—0	0100	Parameter	Extended Store Version
Bytes	2—33	Version label		32 character version string, unused bytes shall be filled with spaces. Only 8-bit Strings are allowed.	

NOTE This message is available in VT version 5 and later.

E.13 Extended Store Version response

The VT acknowledges whether the object pool was stored in the non-volatile storage.

Transmission repetition rate:	In response to Extended Store Version command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 212_{10}			
	Bits	7—4	1101	Command	Non Volatile Memory
	Bits	3—0	0100	Parameter	Extended Store Version
Bytes	2—5			Reserved, set to FF_{16}	
Byte	6			Error Codes (0 = no errors; successfully stored) Bit 0 = Reserved Bit 1 = 1 = Version label is not correct Bit 2 = 1 = Insufficient memory available Bit 3 = 1 = Any other error	
Bytes	7, 8			Reserved, set to FF_{16}	

NOTE This message is available in VT version 5 and later.

E.14 Extended Load Version command

The Extended Load Version command allows a Working Set to load a copy of an object pool from the non-volatile storage of the VT. If an object pool is already loaded it is overwritten. If the message is acknowledged positive by the VT, the Working Set can proceed as if all objects had been transmitted normally. If the version label contains no string (all blanks), the last stored extended version in non-volatile storage shall be loaded.

When the VT receives the Extended Load Version command, it shall set the “parsing” bit in the VT Status message to 1 until it has finished parsing the object pool and sends the Extended Load Version Response message.

The Working Set Master shall wait for the Extended Load Version response until three consecutive VT Status messages have been received where the “parsing” bit is set to 0. At that time, if the Extended Load Version response has not been received by the Working Set Master, then it shall assume that the Extended Load Version command was not received by the VT. Three messages are waited for to avoid race conditions created by a VT Status message that can already be in a transmit queue, which does not correctly identify the parsing state.

Transmission repetition rate:	On request
Data length:	33 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 213_{10}			
	Bits	7—4	1101	Command	Non Volatile Memory
	Bits	3—0	0101	Parameter	Extended Load Version
Bytes	2—33	Version label		32 character version string, unused bytes shall be filled with spaces. Only 8-bit Strings are allowed.	

NOTE This message is available in VT version 5 and later.

E.15 Extended Load Version response

The VT acknowledges whether a copy was loaded from the non-volatile storage.

Transmission repetition rate:	In response to an Extended Load Version command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 213_{10}			
	Bits	7—4	1101	Command	Non Volatile Memory
	Bits	3—0	0101	Parameter	Extended Load Version
Bytes	2—5			Reserved, set to FF_{16}	
Byte	6			Error Codes (0 = no errors; successfully loaded) Bit 0 = 1 = File system error or pool data corruption Bit 1 = 1 = Version label is not correct or Version label unknown Bit 2 = 1 = Insufficient memory available Bit 3 = 1 = Any other error	
Bytes	7, 8			Reserved, set to FF_{16}	

NOTE This message is available in VT version 5 and later.

E.16 Extended Delete Version command

The Extended Delete Version command allows a Working Set to delete an extended version of an object pool in the non-volatile storage of the VT. If a copy of this version is in the volatile memory at the same time it is preserved there — this message affects non-volatile storage only.

If the version label contains no string (all blanks) the last stored extended version in non-volatile storage is to be deleted. Repeating this command causes each successive “last stored version” to be deleted.

If the version label contains a single asterisk character (*) padded right with 31 blanks this shall be interpreted by the VT as a wild card and shall cause the deletion of all extended version object pools in NVM owned by the requesting Working Set. It does not delete object pools owned by other Working Sets[7].

NOTE To delete the object pool from volatile memory, see F.44.

7) The asterisk wild-card delete-all function is available in VT Version 6 and later.

Transmission repetition rate:	On request
Data length:	33 bytes
Parameter group number:	ECU to VT, Destination-Specific

Byte	1	VT function = 214_{10}		
	Bits	7—4 1101	Command	Non Volatile Memory
	Bits	3—0 0110	Parameter	Extended Delete Version
Bytes	2—33	Version label	32 character version string, unused bytes shall be filled with spaces. Only 8-bit Strings are allowed.	

NOTE This message is available in VT version 5 and later.

E.17 Extended Delete Version response

The VT acknowledges whether a version was deleted in the non-volatile storage.

Transmission repetition rate:	In response to Extended Delete Version command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific

Byte	1	VT function = 214_{10}		
	Bits	7—4 1101	Command	Non Volatile Memory
	Bits	3—0 0110	Parameter	Extended Delete Version
Bytes	2—5		Reserved, set to FF_{16}	
Byte	6		Error Codes (0 = no errors; successfully deleted or wild card delete was commanded) Bit 0 = Reserved Bit 1 = 1 = Version label is not correct or Version label unknown Bit 2 = Reserved Bit 3 = 1 = Any other error	
Bytes	7, 8		Reserved, set to FF_{16}	

NOTE This message is available in VT version 5 and later.

Annex F
(normative)

Command and Macro messages

F.1 General

Commands are sent from a Working Set to the VT using the PGNs given in Annex C. Working Set Masters or members can send any command in this annex. The VT shall respond to these commands even if no object pool of the originating Working Set is loaded. The originator shall wait for a response before sending another command. Unless stated otherwise, another command can be sent if a response is not received within 1,5 s. The VT design could decouple the responses to command messages from the display refresh to avoid a response timeout at the working set. But the VT designer should be aware this could lead to synchronization problems between the working set and the VT. The Working Set designer can choose to resend the previous command (typically not more than 3 times). Care should be taken in what command is being repeated (e.g. repeating “Change Child Location” could cause a cumulative visual change). Each of the commands in this annex can also be used in a Macro unless otherwise noted.

Unless otherwise noted any attribute in a response message which also exists in the command message shall be set to the same value as in the command message, i.e. the response frame reflects the command but not necessarily the state of the object.

Example An Enable/Disable Object command is sent with Object Id = 11000 and Byte 4 = 0 (disable). The object is currently in a state where it cannot be disabled, and therefore it stays enabled. The response frame is sent with Object Id = 11000 and Byte 4 = 0 (disable) and Error Code indicating the cause of the error.

F.2 Hide/Show Object command

The Hide/Show Object command is used to hide or show a Container object. This pertains to the visibility of the object as well as its remembered state.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 160_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	0000	Parameter	Hide/Show object
Bytes	2, 3			Object ID	
Byte	4			0 = Hide, 1 = Show	
Bytes	5—8			Reserved, set to FF_{16}	

F.3 Hide/Show Object response

The VT uses this message to respond to the Hide/Show Object command.

Transmission repetition rate:	In response to Hide/Show Object command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 160_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	0000	Parameter	Hide/Show object
Bytes	2, 3			Object ID	
Byte	4			0 = Hide, 1 = Show	
Byte	5			Error Codes (0 = no errors) Bit 0 = 1 = References to missing child objects[a] Bit 1 = 1 = Invalid Object ID Bit 2 = 1 = Command error Bit 3 = undefined, set to 0 recommended Bit 4 = 1 = Any other error	
Bytes	6—8			Reserved, set to FF_{16}	

[a] The VT design can support immediate reporting of references to missing objects, or the VT design can defer the validation until the Container, and its child objects are shown, when it can be reported with the VT Change Active Mask message.

F.4 Enable/Disable Object command

This command is used to enable or disable an input field object or a Button object and pertains to the accessibility of an input field object or Button object. This command is also used to enable or disable an Animation object.

It is allowed to enable already enabled objects and to disable already disabled objects. If this happens the response frame shall indicate "no errors" and macros shall be executed.

NOTE VT version 3 and prior do not support enable/disable of a Button object.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 161_{10}		
	Bits	7—4 1010	Command	Command
	Bits	3—0 0001	Parameter	Enable/Disable object
Bytes	2, 3		Object ID	
Byte	4		Enable/Disable command 0 = Disable, 1 = Enable	
Bytes	5—8		Reserved, set to FF_{16}	

F.5 Enable/Disable Object response

The VT uses this message to respond to the Enable/Disable Object command.

Transmission repetition rate:	In response to an Enable/Disable Object command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 160_{10}		
	Bits	7—4 1010	Command	Command
	Bits	3—0 0001	Parameter	Enable/Disable Object
Bytes	2, 3		Object ID	
Byte	4		0 = Disable, 1 = Enable	
Byte	5		Error Codes (0 = no errors) Bit 0 = Undefined, set to 0 recommended Bit 1 = 1 = Invalid Object ID Bit 2 = 1 = Invalid Enable/Disable command value Bit 3 = 1 = Could not complete. This input object is currently being modified. Bit 4 = 1 = Any other error	
Bytes	6—8		Reserved, set to FF_{16}	

F.6 Select Input Object command

This command is used to force the selection of an input field, Button, or Key object. The VT shall provide a way for the operator to recognize a selected object. If the object is disabled or not visible, an error code is returned. Depending on byte 4, the object is either selected (has focus) or opened for input (not valid for Button objects or Key objects).

If the object to be selected is included multiple times on the same mask, it is proprietary to the VT which of the object instances will be selected (has focus).

NOTE 1 Even if the input field is activated for data input, the value is not changed by this command (e.g. Input Boolean is not toggled).

NOTE 2 This command originates in the Working Set and is a command to the VT. In the situation where the change originates with the operator, the VT indicates the selected input object with the VT Select Input Object message (see H.8).

NOTE 3 VT version 3 and prior do not support selection of a Button object or a Key object.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 162_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	0010	Parameter	Select Input object
Bytes	2, 3			Object ID — NULL indicates that no object shall be selected (i.e. focus is removed), Option byte shall be FF_{16} when Object ID is NULL	
Byte	4			Option FF_{16} = Set Focus to object referenced by Object ID or remove focus if Object ID is the NULL object 0 = Activate for data-input the object referenced by Object ID (invalid for Button object or Key object and can have no effect for the Input Boolean object depending on the VT design) NOTE Value 0 available only on VT Version 4 and later.	
Bytes	5—8			Reserved, set to FF_{16}	

F.7 Select Input Object response

The VT uses this message to respond to the Select Input Object command.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 162_{10}		
	Bits	7—4 1010	Command	Command
	Bits	3—0 0010	Parameter	Select Input object
Bytes	2, 3		Object ID	
Byte	4		Response 0 = Object referenced by Object ID is not selected or Object ID is the NULL object or Byte 5 indicates an error 1 = Object referenced by Object ID is Selected 2 = Object referenced by Object ID is Opened for Edit[a]	
Byte	5		Error Codes (0 = no errors) Bit 0 = 1 = Object is disabled Bit 1 = 1 = Invalid Object ID Bit 2 = 1 = Object is not on the active mask or object is in a hidden container Bit 3 = 1 = Could not complete. Another Input field is currently being modified, or a Button or Soft Key is currently being held. Bit 4 = 1 = Any other error Bit 5 = 1 = Invalid Option value[a] NOTE An object that is off-screen or zero width or zero height, and is enabled and not within a hidden container, is both selectable and able to be opened for edit. This is not an error condition.	
Bytes	6—8		Reserved, set to FF_{16}	

[a] These bits/values exist in VT version 4 and later.

F.8 ESC command

This command is used to abort operator input.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 146_{10}		
	Bits	7—4 1001	Command	Command
	Bits	3—0 0010	Parameter	ESC
Bytes	2—8		Reserved, set to FF_{16}	

F.9 ESC response

The VT uses this message to respond to the ESC command.

Transmission repetition rate:	In response to ESC command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 146_{10}		
	Bits	7—4 1001	Command	Command
	Bits	3—0 0010	Parameter	ESC
Bytes	2, 3		Object ID where input was aborted if no error code	
Byte	4		Error Codes (0 = no errors) Bit 0 = 1 = No input field is open for input, ESC ignored. Bits 1—3 = Undefined, set to 0 recommended Bit 4 = 1 = Any other error	
Bytes	5—8		Reserved, set to FF_{16}	

F.10 Control Audio Signal command

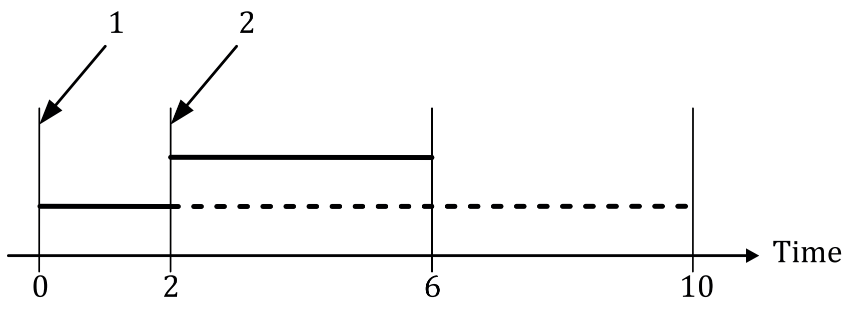
This command can be used to control the audio on the VT. When received this message shall terminate any audio in process from the originating Working Set and replace the previous command with the new command. The previous rule does not apply to an acoustic signal associated with an Alarm Mask. There can be a momentary interruption in the tone while the VT terminates the previous command.

Continuous tones are not recommended, however it is recognized that 255 activations of 65,535 ms each produces 278 min of continuous tone. If this is insufficient, the originating Working Set can issue an additional Control Audio Signal command to the VT prior to the expiration of the tone.

If the VT is capable of supporting the Control Audio Signal command for only a single Working Set at a time, and if it is currently processing a command from a different Working Set, then the Control Audio Signal response shall indicate that the Audio Device is busy.

If the VT is capable of simultaneously supporting the Control Audio Signal command from more than one Working Set ("multisound"), then the command is accepted up to the limit of the VT capabilities.

The audio produced by a control audio command shall never be queued or delayed beyond normal VT message processing. See Figure F.1.



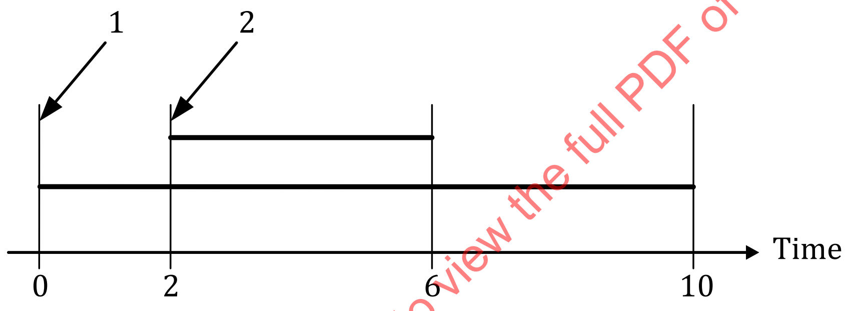
Key

1 Working Set 1 sends Control Audio Device command with a total time of 10 s

2 Working Set 2 alarm becomes visible and has acoustic signal other than "none". VT terminates Working Set 1 audio and informs Working Set 1. VT proprietary acoustic requires 4 s

Figure F.1 — Acoustic signal termination (multisound not supported)

If the VT is capable of supporting the Control Audio Signal command for more than a single Working Set at a time, then the audio for each Working Set is not interrupted. See Figure F.2.



Key

1 Working Set 1 sends Control Audio Device command with a total time of 10 s

2 Working Set 2 alarm becomes visible and has acoustic signal other than "none". VT proprietary acoustic requires 4 s

Figure F.2 — Acoustic signal with multisound

NOTE The Control Audio Signal command is independent of the currently active Working Set, therefore audio tones can be commanded even if the originating Working Set is not the active Working Set (see 4.6.14).

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 163_{10}		
	Bits	7—4 1010	Command	Command
	Bits	3—0 0011	Parameter	Control Audio
Byte	2		Activations 0 = Terminates any audio in process from the originating Working Set (Frequency and Duration values are ignored). 1—255 = Number of Audio Activations.	
Bytes	3, 4		Frequency in Hz If the Frequency specified is outside of the VT capabilities for production of sound (also applies to non-multiple frequency devices) then the VT limits the frequency to the reproducible range.	
Bytes	5, 6		On-time duration in ms. If the duration specified is non-zero and less than the VT capabilities for timing, the VT shall time the audio to the VTs smallest controlled value.	
Bytes	7, 8		Off-time duration in ms. If the duration specified is less than the VT capabilities for timing, the VT shall time the audio to the VTs smallest controlled value. If the On-time is zero, this value is ignored and zero is used.	

F.11 Control Audio Signal response

This message is sent by the VT in response to a control audio command.

Transmission repetition rate:	In response to Control Audio Signal command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 163_{10}		
	Bits	7—4 1010	Command	Command
	Bits	3—0 0011	Parameter	Control Audio
Byte	2		Error Codes (0 = no errors) Bit 0 = 1 = Audio device is busy Bits 1-3 = Undefined, set to 0 recommended Bit 4 = 1 = Any other error	
Bytes	3—8		Reserved, set to FF_{16}	

F.12 Set Audio Volume command

This command can be used to control the audio on the VT.

This command applies to subsequent Control Audio Signal commands (see F.10) of the issuing Working Set. This command should also affect the currently playing tone, if any. VTs that are not able to modify the volume of the currently playing tone shall set the Audio device is busy bit in the response. This command should not affect in any way the volume settings of other Working Sets and shall not affect the volume of Alarm Masks.

In VT version 4 and prior, the default audio volume was undefined.

In VT version 5 and later, the default audio volume is 100 % of maximum volume set by the operator.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 164_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	0100	Parameter	Set Audio Volume
Byte	2			Percent (0 % to 100 %) of maximum volume set by operator	
Bytes	3—8			Reserved, set to FF_{16}	

F.13 Set Audio Volume response

This message is sent by the VT in response to a Set Audio Volume command.

Transmission repetition rate:	In response to Set Audio Volume command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 164_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	0100	Parameter	Set Audio Volume
Byte	2			Error Codes (0 = no error) Bit 0 = 1 = Audio device is busy, commands use the new setting Bit 1 = 1 = Command is not supported[a] Bits 2-3 = Undefined, set to 0 recommended Bit 4 = 1 = Any other error	
Bytes	3—8			Reserved, set to FF_{16}	

[a] This bit exists in VT version 5 and later.

F.14 Change Child Location command

The Change Child Location command is used to change the position of an object. The new position is set relative to the object's current position. Since the object can be included in many parent objects, the parent Object ID is also included. If a parent object includes the child object multiple times, then each instance will be moved. When the object is moved, the parent object shall be refreshed. The position attributes given in the message have an offset of −127 (i.e. value of 0 = a − 127 pixel move, 255 = a + 128 pixel move). Positive values indicate a position change down (Y) or to the right (X). Negative values indicate a position change up (Y) or to the left (X).

Because of the possibility of lost messages, when a guaranteed position is required, the Change Child Position command should be used instead of specifying relative coordinates with the Change Child Location command.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 165_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	0101	Parameter	Change Child Location
Bytes	2, 3			Parent Object ID	
Bytes	4, 5			Object ID of object to move	
Byte	6			Relative change in X position	
Byte	7			Relative change in Y position	
Byte	8			Reserved, set to FF_{16}	

F.15 Change Child Location response

This message is sent by the VT in response to a Change Child Location command.

Transmission repetition rate:	In response to Change Child Location command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 165_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	0101	Parameter	Change Child Location
Bytes	2, 3			Parent Object ID	

Bytes	4, 5		Object ID of object to move	
Byte	6		Error Codes (0 = no error) Bit 0 = 1 = The referenced parent object ID does not exist, or is not a parent of the target object Bit 1 = 1 = The target object does not exist, or this command is not applicable to the target object Bit 2, 3 = Undefined, set to 0 recommended Bit 4 = 1 = Any other error	
Bytes	7, 8		Reserved, set to FF_{16}	

F.16 Change Child Position command

The Change Child Position command is used to change the position of an object. The new position is set relative to the parent object's position. Since the object can be included in many parent objects, the parent Object ID is also included. If a parent object includes the child object multiples times, then each instance will be moved to the same location (the designer can want to use Change Child Location command to move all instances in a relative motion). When the object is moved, the parent object shall be refreshed. The position attributes given in the message are signed integer. Positive values indicate a position below (Y) or to the right of (X) the top left corner of the parent object. Negative values indicate a position above (Y) or to the left of (X) the top left corner of the parent object.

Transmission repetition rate:	On request
Data length:	9 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 180_{10}		
	Bits	7—4 1011	Command	Command
	Bits	3—0 0100	Parameter	Change Child Position
Bytes	2, 3		Parent Object ID	
Bytes	4, 5		Object ID of object to move	
Bytes	6, 7		New X position relative to the top left corner of parent object.	
Bytes	8, 9		New Y position relative to the top left corner of parent object.	

F.17 Change Child Position response

This message is sent by the VT in response to the Change Child Position command.

Transmission repetition rate:	In response to Change Child Position command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 180_{10}		
	Bits	7—4 1011	Command	Command
	Bits	3—0 0100	Parameter	Change Child Position
Bytes	2, 3		Parent Object ID	
Bytes	4, 5		Object ID of object to move	
Byte	6		Error Codes (0 = no error) Bit 0 = 1 = The referenced parent object ID does not exist, or is not a parent of the target object Bit 1 = 1 = The target object does not exist, or this command is not applicable to the target object Bit 2,3 = Undefined, set to 0 recommended Bit 4 = 1 = Any other error	
Bytes	7, 8		Reserved, set to FF_{16}	

F.18 Change Size command

The Change Size command is used to change the size of an object. A value of 0 for width or height or both means that the object size is 0 and the object is not drawn.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 166_{10}		
	Bits	7—4 1010	Command	Command
	Bits	3—0 0110	Parameter	Change Size
Bytes	2, 3		Object ID of object to size	
Bytes	4, 5		New width	
Bytes	6, 7		New height	
Byte	8		Reserved, set to FF_{16}	

F.19 Change Size response

This message is sent by the VT in response to a Change Size command.

Transmission repetition rate:	In response to Change Size command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 166_{10}			
	Bits	7—4 1010	Command	Command	
	Bits	3—0 0110	Parameter	Parameter	Change Size
Bytes	2, 3		Object ID of object to size		
Byte	4		Error Codes (0 = no error) Bit 0 = 1 = Invalid Object ID Bit 1 = Undefined, set to 0 recommended Bit 2 = Undefined, set to 0 recommended Bit 3 = Undefined, set to 0 recommended Bit 4 = 1 = Any other error		
Bytes	5—8		Reserved, set to FF_{16}		

F.20 Change Background Colour command

This command is used to change the background colour of an object.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 167_{10}		
	Bits	7—4 1010	Command	Command
	Bits	3—0 0111	Parameter	Change Background Colour
Bytes	2, 3		Object ID of object to change	
Byte	4		New Background colour (see A.3)	
Bytes	5—8		Reserved, set to FF_{16}	

F.21 Change Background Colour response

This message is sent by the VT in response to a Change Background Colour command.

Transmission repetition rate:	In response to Change Background Colour command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 167_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	0111	Parameter	Change Background Colour
Bytes	2, 3			Object ID	
Byte	4			New Background colour (see A.3)	
Byte	5			Error Codes (0 = no error) Bit 0 = 1 = Invalid Object ID Bit 1 = 1 = Invalid colour code Bits 2—3 = Undefined, set to 0 recommended Bit 4 = 1 = Any other error	
Bytes	6—8			Reserved, set to FF_{16}	

F.22 Change Numeric Value command

This command is used to change the value of an object. It applies only to objects that have a numeric "value" attribute. The size of the object shall not be changed by this command. Only the object indicated in the command is to be changed, variables referenced by the object are not changed.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 168_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	1000	Parameter	Change Numeric Value command
Bytes	2, 3			Object ID of object to change	
Byte	4			Reserved, set to FF_{16}	

Bytes	5—8	New value for value attribute. Size depends on object type. Objects of size 1 byte are found in byte 5. Objects of size 2 bytes are found in bytes 5—6. Values greater than 1 byte are transmitted little endian (LSB first). Unused bytes shall be filled with zero.

Input Boolean object:	1 byte for TRUE/FALSE
Input Number object:	4 bytes for integer input
Input List object:	1 byte for list index
Output Number object:	4 bytes for integer out-
Output List object[a]:	put
Output Meter object:	1 byte for list index
Output Linear Bar	2 bytes for integer value
Graph object:	2 bytes for integer value
Output Arched Bar	
Graph object:	2 bytes for integer value
Number Variable	
object:	4 bytes for integer value
Object Pointer object:	
External Object Pointer	2 bytes for Object ID
object[b]:	Bytes 5—6: External Reference NAME Object ID
	Bytes 7—8: Referenced
Animation object[b]:	Object ID
Scaled Graphic object[c]:	1 byte for list index
	2 bytes for Object ID

[a] VT version 4 and later.

[b] VT version 5 and later.

[c] VT version 6 and later.

The frequency of update is at the discretion the Working Set designer; however the designer should consider the limited bandwidth available (see 4.6.10.1).

F.23 Change Numeric Value response

The VT sends this message in response to the Change Numeric Value command.

Transmission repetition rate:	In response to Change Numeric Value command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 168_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	1000	Parameter	Change Numeric Value command
Bytes	2, 3			Object ID	

Byte	4	Error Codes (0 = no error) Bit 0 = 1 = Invalid Object ID Bit 1 = 1 = Invalid value[c] Bit 2 = Reserved, set to 0[d] Bit 3 = Undefined, set to 0 recommended Bit 4 = 1 = Any other error
Bytes	5—8	Value. Size depends on object type. Objects of size 1 byte are found in byte 5. Objects of size 2 bytes are found in bytes 5—6. Values greater than 1 byte are transmitted little endian (LSB first): Input Boolean object: 1 byte for TRUE/FALSE Input Number object: 4 bytes for integer input Input List object: 1 byte for list index Output Number object: 4 bytes for integer output Output List object[a]: 1 byte for list index Output Meter object: 2 bytes for integer value Output Linear Bar Graph object: 2 bytes for integer value Output Arched Bar Graph object: 2 bytes for integer value Number Variable object: 4 bytes for integer value Object Pointer object: 2 bytes for Object ID External Object Pointer object[b]: Bytes 5—6: External Reference NAME Object ID Bytes 7—8: Referenced Object ID Animation object[b]: 1 byte for list index Scaled Graphic object[e]: 2 bytes for Object ID

[a] VT version 4 and later.

[b] VT version 5 and later.

[c] This bit is only set when the Change Numeric Value command is used to change a pointer value to an invalid object.

[d] In VT version 4 and 5, this bit indicated “1 = Value in use (e.g. open for input)”. This behaviour is deprecated in VT version 6 and later (see 4.6.10.2).

[e] VT version 6 and later.

F.24 Change String Value command

This command is used to change the value of an object. It applies only to objects that have a string “value” attribute. The size of the object shall not be changed by this command. Only the object indicated in the command is to be changed, variables referenced by the object are not changed.

If the message contents fit in a single packet, transport protocol shall not be used. If the transferred string has a length of 3 bytes or less, the remaining bytes in the single packet message shall be set to FF_{16}.

The transferred string is allowed to be smaller than the length of the value attribute of the target object and in this case the VT shall pad the value attribute with space characters. The number of bytes in the transfer string (Bytes 4,5) shall be less than or equal to the length attribute of the target object (i.e. string length shall not be increased).

Transmission repetition rate:	On request
Data length:	Variable
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 179_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	0011	Parameter	Change String Value
Bytes	2, 3			Object ID of the object to change	
Bytes	4, 5			Total number of bytes in the string to transfer (bytes to follow)	
Bytes	6—n			New string value	

F.25 Change String Value response

This message is sent by the VT in response to the Change String Value message.

Transmission repetition rate:	In response to Change String Value message
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 179_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	0011	Parameter	Change String Value
Bytes	2, 3			Reserved, set to FF_{16}	
Bytes	4, 5			Object ID of the object to change	
Byte	6			Error Codes (0 = no error) Bit 0 = Undefined, set to 0 recommended Bit 1 = 1 = Invalid Object ID Bit 2 = 1 = String too long Bit 3 = 1 = Any other error Bit 4 = 1 = Reserved, set to 0[a]	
Bytes	7—8			Reserved, set to FF_{16}	

[a] In VT version 4 and 5, this bit indicated “1 = Value in use (e.g. open for input)”. This behaviour is deprecated in VT version 6 and later (see 4.6.10.2).

F.26 Change End Point command

This command is used to change the end point of an Output Line object by changing the width, height and/or line direction attributes.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 169_{10}		
	Bits	7—4 1010	Command	Command
	Bits	3—0 1001	Parameter	Change End Point
Bytes	2, 3		Object ID of an Output Line object to change	
Bytes	4, 5		Width in pixels	
Bytes	6, 7		Height in pixels	
Bytes	8		Line Direction (refer to Output Line object attributes)	

F.27 Change End Point response

The VT uses this message to respond to the Change End Point command.

Transmission repetition rate:	In response to Change End Point command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 169_{10}		
	Bits	7—4 1010	Command	Command
	Bits	3—0 1001	Parameter	Change End Point
Bytes	2, 3		Object ID	
Byte	4		Error Codes (0 = no error) Bit 0 = 1 = Invalid Object ID Bit 1 = 1 = Invalid Line Direction Bit 2 = Undefined, set to 0 recommended Bit 3 = Undefined, set to 0 recommended Bit 4 = 1 = Any other error	
Bytes	5—8		Reserved, set to FF_{16}	

F.28 Change Font Attributes command

This command is used to change the Font Attributes in a Font Attributes object.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 170_{10}				
	Bits	7—4	1010	Command	Command	
	Bits	3—0	1010	Parameter	Change Font Attributes	
Bytes	2, 3			Object ID of object to change		
Byte	4			Font colour (see A.3)		
Byte	5			Font size		
Byte	6			Font type		
Byte	7			Font style		
Byte	8			Reserved, set to FF_{16}		

F.29 Change Font Attributes response

The VT uses this message to respond to the Change Font Attributes command.

Transmission repetition rate:	In response to the Change Font Attributes command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 170_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	1010	Parameter	Change Font attributes
Bytes	2, 3			Object ID	
Byte	4			Error Codes (0 = no error) Bit 0 = 1 = Invalid Object ID Bit 1 = 1 = Invalid colour Bit 2 = 1 = Invalid size Bit 3 = 1 = Invalid type Bit 4 = 1 = Invalid style Bit 5 = 1 = Any other error	
Bytes	5—8			Reserved, set to FF_{16}	

F.30 Change Line Attributes command

This command is used to change the Line Attributes in a Line Attributes object.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 171_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	1011	Parameter	Change Line Attributes
Bytes	2, 3			Object ID of object to change	
Byte	4			Line Colour (see A.3)	
Byte	5			Line Width	
Bytes	6, 7			Line Art	
Bytes	8			Reserved, set to FF_{16}	

F.31 Change Line Attributes response

The VT uses this message to respond to the Change Line Attributes command.

Transmission repetition rate:	In response to the Change Line Attributes command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 171_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	1011	Parameter	Change Line Attributes
Bytes	2, 3			Object ID	
Byte	4			Error Codes (0 = no error) Bit 0 = 1 = Invalid Object ID Bit 1 = 1 = Invalid colour Bit 2 = 1 = Invalid width Bit 3 = Undefined, set to 0 recommended Bit 4 = 1 = Any other error	
Bytes	5—8			Reserved, set to FF_{16}	

F.32 Change Fill Attributes command

This command is used to change the Fill Attributes in a Fill Attributes object.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 172_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	1100	Parameter	Change Fill Attributes
Bytes	2, 3			Object ID of object to change	
Byte	4			Fill Type	
Byte	5			Fill Colour (see A.3)	
Bytes	6, 7			Fill Pattern Object ID	
Byte	8			Reserved, set to FF_{16}	

F.33 Change Fill Attributes response

The VT uses this message to respond to the Change Fill Attributes command.

Transmission repetition rate:	In response to Change Fill Attributes command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 172_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	1100	Parameter	Change Fill Attributes
Bytes	2, 3			Object ID	
Byte	4			Error Codes (0 = no error) Bit 0 = 1 = Invalid Object ID Bit 1 = 1 = Invalid type Bit 2 = 1 = Invalid colour Bit 3 = 1 = Invalid pattern Object ID Bit 4 = 1 = Any other error	
Bytes	5—8			Reserved, set to FF_{16}	

F.34 Change Active Mask command

This command is used to change the active mask of a Working Set to either a Data Mask object or an Alarm Mask object.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 173_{10}		
	Bits	7—4 1010	Command	Command
	Bits	3—0 1101	Parameter	Change Active Mask
Bytes	2, 3		Working Set Object ID	
Bytes	4, 5		New Active Mask Object ID	
Bytes	6—8		Reserved, set to FF_{16}	

F.35 Change Active Mask response

The VT uses this message to respond to the Change Active Mask command (see H.14).

Transmission repetition rate:	In response to Change Active Mask command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 173_{10}		
	Bits	7—4 1010	Command	Command
	Bits	3—0 1101	Parameter	Change Active Mask
Bytes	2, 3		New Active Mask Object ID	
Byte	4		Error Codes (0 = no error) Bit 0 = 1 = Invalid Working Set Object ID Bit 1 = 1 = Invalid Mask Object ID Bit 2 = Undefined, set to 0 recommended Bit 3 = Undefined, set to 0 recommended Bit 4 = 1 = Any other error	
Bytes	5—8		Reserved, set to FF_{16}	

F.36 Change Soft Key Mask command

This command is used to change the Soft Key Mask associated with a Data Mask object or an Alarm Mask object.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 174_{10}		
	Bits	7—4 1010	Command	Command
	Bits	3—0 1110	Parameter	Change Soft Key Mask
Byte	2		Mask Type (1 = Data, 2 = Alarm)	
Bytes	3, 4		Data or Alarm Mask Object ID	
Bytes	5, 6		New Soft Key Mask Object ID	
Bytes	7, 8		Reserved, set to FF_{16}	

F.37 Change Soft Key Mask response

The VT uses this message to respond to the Change Soft Key Mask command (see also H.16).

Transmission repetition rate:	In response to Change Soft Key Mask command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 174_{10}		
	Bits	7—4 1010	Command	Command
	Bits	3—0 1110	Parameter	Change Soft Key Mask
Bytes	2, 3		Data or Alarm Mask Object ID	
Bytes	4, 5		New Soft Key Mask Object ID	
Byte	6		Error Codes (0 = no error) Bit 0 = 1 = Invalid Data or Alarm Mask Object ID Bit 1 = 1 = Invalid Soft Key Mask Object ID Bit 2 = 1 = Missing Objects Bit 3 = 1 = Mask or child object has errors Bit 4 = 1 = Any other error	
Bytes	7, 8		Reserved, set to FF_{16}	

F.38 Change Attribute command

This command is used to change any attribute with an assigned AID. This message cannot be used to change strings.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 175_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	1111	Parameter	
Bytes	2, 3			Change Attribute	
Byte	4			Attribute ID (AID)	
Bytes	5—8			New value for attribute. Size depends on attribute data type. Values greater than 1 byte are transmitted little endian (LSB first). Unused bytes should be set to zero: Boolean: 1 byte for TRUE/FALSE Integer: 1, 2 or 4 bytes as defined in object tables Float: 4 bytes Bitmask: 1 byte	

F.39 Change Attribute response

The VT uses this message to respond to the Change Attribute command.

Transmission repetition rate:	In response to Change Attribute command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 175_{10}			
	Bits	7—4	1010	Command	Command
	Bits	3—0	1111	Parameter	Change Attribute
Bytes	2, 3			Object ID	
Byte	4			Attribute ID (AID)	
Byte	5			Error Codes (0 = no error) Bit 0 = 1 = Invalid Object ID Bit 1 = 1 = Invalid Attribute ID Bit 2 = 1 = Invalid value Bit 3 = 1 = Reserved, set to 0[a] Bit 4 = 1 = Any other error	
Bytes	6—8			Reserved, set to FF_{16}	

[a] In VT version 4 and 5, this bit indicated “1 = Value in use (e.g. open for input)”. This behaviour is deprecated in VT version 6 and later (see 4.6.10.2).

F.40 Change Priority command

This command is used to change the priority of an Alarm Mask. This command causes the VT to evaluate the priority of all active masks and can cause a change to a different mask if the Alarm Mask being changed should either become the active Working Set and mask, or should no longer be the active Working Set and mask.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 176_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	0000	Parameter	Change Priority
Bytes	2, 3			Object ID of Alarm Mask	
Byte	4			New priority	
Bytes	5—8			Reserved, set to FF_{16}	

F.41 Change Priority response

The VT uses this message to respond to the Change Priority command.

Transmission repetition rate:	In response to Change Priority
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 176_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	0000	Parameter	Change Priority
Bytes	2, 3			Object ID of Alarm MaskByte	
Byte	4			New priority	
Byte	5			Error Codes (0 = no error) Bit 0 = 1 = Invalid Object ID Bit 1 = 1 = Invalid priority Bits 2-3 = Undefined, set to 0 recommended Bit 4 = 1 = Any other error	
Bytes	6—8			Reserved, set to FF_{16}	

F.42 Change List Item command

This command is used to change a list item in an Input List object, Output List object[8], Animation object[9], or External Object Definition object[9].

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 177_{10}				
	Bits	7—4	1011	Command	Command	
	Bits	3—0	0001	Parameter	Change List Item	
Bytes	2, 3			Object ID of an Input List object, Output List object, Animation object, or External Object Definition object		
Byte	4			List Index (items are numbered 0—n)		
Bytes	5, 6			New Object ID or $FFFF_{16}$ to set empty item		
Bytes	7, 8			Reserved, set to FF_{16}		

F.43 Change List Item response

The VT uses this message to respond to the Change List Item command.

Transmission repetition rate:	In response to Change List Item command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 177_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	0000	Parameter	Change List Item
Bytes	2, 3			Object ID of an Input List object or Output List object[a], Animation object[b], or External Object Definition object[b]	
Byte	4			List Index (items are numbered 0—n)	
Bytes	5, 6			New Object ID or $FFFF_{16}$ to set empty item	

8) VT version 4 and later.

9) VT version 5 and later.

Byte	7	Error Codes (0 = no error) Bit 0 = 1 = Invalid Input List object ID or Output List object ID[a], Animation object[b], External Object Definition object[b] Bit 1 = 1 = Invalid List Index Bit 2 = 1 = Invalid New List Item Object ID Bit 3 = Reserved, set to 0[c] Bit 4 = 1 = Any other error
Byte	8	Reserved, set to FF_{16}

a VT version 4 and later.

b VT version 5 and later.

c In VT version 4 and 5, this bit indicated "1 = Value in use (e.g. open for input)". This behaviour is deprecated in VT version 6 and later (see 4.6.10.2).

F.44 Delete Object Pool command

This command is used to delete the entire object pool of this Working Set from volatile storage. This command can be used by an implement when it wants to move its object pool to another VT, or when it is shutting down or during the development of object pools.

NOTE To delete an object pool from non-volatile storage in the VT; see E.8.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 178_{10}			
	Bits	7–4	1011	Command	Command
	Bits	3–0	0010	Parameter	Delete Object Pool
Bytes	2–8			Reserved, set to FF_{16}	

F.45 Delete Object Pool response

The VT uses this message to respond to the Delete Object Pool command.

Transmission repetition rate:	In response to Delete Object Pool command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 178_{10}		
	Bits	7—4 1011	Command	Command
	Bits	3—0 0010	Parameter	Delete Object Pool
Byte	2		Error Codes (0 = no error, deletion was successful) Bit 0 = 1 = Deletion error Bits 1-3 = Undefined, set to 0 recommended Bit 4 = 1 = Any other error	
Bytes	3—8		Reserved, set to FF_{16}	

F.46 Lock/Unlock Mask command

This command is used by a Working Set to disallow or allow screen refreshes at the VT for the visible Data Mask or User-Layout Data Mask owned by the requesting Working Set. This message would be used when a series of changes need to be synchronized or made visually atomic (for example during animation). A Lock command does not imply that drawing stops, only that changes to the visible mask are not made visible to an operator until one of the unlock mechanisms listed below occurs:

— An Unlock command is received and the visible mask has been refreshed.

— A timeout occurs based on the timeout attribute of the Lock message.

— Navigation to, or activation of input objects or Buttons on the Data Mask.

— A change from visible to hidden mask occurs.

— The pool is deleted.

— A proprietary reason (e.g. an input dialogue closes).

When a mask is locked, the on screen presentation of the mask is not updated for any reason. This includes flashing objects and any animation object that is on the mask. If the Animation object is a timed animation, the timer will continue to run normally in the background, the animation object will be updated on the non-visible copy of the mask, but the on screen presentation is not updated until the mask is unlocked.

While locked, CAN messages/commands, key and button presses, events and macros are still processed normally. When one of the unlock mechanisms occurs, a response message is sent and normal periodic screen refreshes resume. The lock state does not apply to Soft Key Masks and Alarm Masks which shall be displayed regardless.

If an Alarm Mask from any Working Set is active when the lock command is received, the lock command is rejected if the active Alarm Mask is in the same display area.

The VT shall respond as soon as possible to a Lock Mask command. The VT's response to the Unlock command depends on whether or not a Data Mask or User-Layout Data Mask is visible. If a Data Mask or User-Layout Data Mask is hidden (e.g. VT is displaying a home page, setup screen, etc.), the VT shall respond to the Unlock command immediately and indicate that the command was ignored. However, if a Data Mask or User-Layout Data Mask is visible, the VT shall not respond to the Unlock Mask command until the Data Mask or User-Layout Data Mask has been completely refreshed (all changes during lock made visible to the operator). If a timeout occurs or a change causing the current Data Mask or User-Layout Data Mask to be hidden, the VT shall send an unsolicited Lock/Unlock Mask Response message with appropriate error codes set.

Typically the Working Set would lock the mask, send any necessary change commands, unlock the mask and then wait for the Lock/Unlock Mask Response message. This will allow mask changes to be synchronized and made visually atomic. To avoid operator interface lags, navigation problems and timing fluctuations in flashing objects, it is recommended that locks be applied for very short periods of time, likely measured in (but not limited to) milliseconds.

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 189_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	1101	Parameter	Lock/Unlock Mask
Byte	2			Command: 0 = Unlock Data Mask or User-Layout Data Mask 1 = Lock Data Mask or User-Layout Data Mask	
Bytes	3, 4			Object ID of the Data Mask or User-Layout Data Mask to Lock or Unlock. If this does not match the visible mask, the command fails with a response code.	
Bytes	5, 6			Lock timeout in ms or zero for no timeout. Once this period expires, the VT shall automatically release the lock if the Working Set has not done so. This attribute does not apply to an Unlock command.	
Bytes	7, 8			Reserved, set to FF_{16}	

F.47 Lock/Unlock Mask response

The VT uses this message to respond to the Lock/Unlock Mask command or to send an unsolicited message with the reasons given in the Lock/Unlock Mask command (see F.46).

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	In response to Lock/Unlock Mask command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 189_{10}		
	Bits	7—4 1011	Command	Command
	Bits	3—0 1101	Parameter	Lock/Unlock Mask
Byte	2		Command: 0 = Unlock Data Mask or User-Layout Data Mask 1 = Lock Data Mask or User-Layout Data Mask	
Byte	3		Error Codes (0 = no error) Bit 0 = 1 = Command ignored, no mask is visible or given Object ID does not match the visible mask Bit 1 = 1 = Lock command ignored, already locked Bit 2 = 1 = Unlock command ignored, not locked Bit 3 = 1 = Lock command ignored, an Alarm Mask is active Bit 4 = 1 = Unsolicited unlock, timeout occurred Bit 5 = 1 = Unsolicited unlock, this mask is hidden Bit 6 = 1 = Unsolicited unlock, operator induced, or any other error Bit 7 = 1 = Any other error	
Bytes	4—8		Reserved, set to FF_{16}	

F.48 Execute Macro command

This command is used to execute a Macro.

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 190_{10}		
	Bits	7—4 1011	Command	Command
	Bits	3—0 1110	Parameter	Execute Macro
Byte	2		Object ID of Macro object	
Bytes	3—8		Reserved, set to FF_{16}	

F.49 Execute Macro response

The VT uses this message to respond to the Execute Macro command.

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	In response to Execute Macro command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 190_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	1110	Parameter	Execute Macro
Byte	2			Object ID of Macro object	
Byte	3			Error Codes (0 = no error) Bit 0 = 1 = Object ID does not exist Bit 1 = 1 = Object ID is not a Macro object Bit 2 = 1 = Any other error	
Bytes	4—8			Reserved, set to FF_{16}	

F.50 Change Object Label command

This command is used by a Working Set to change a label of an object. See also B.21.

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 181_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	0101	Parameter	Change Object Label
Bytes	2, 3			Object ID of object to associate label with	

Bytes	4, 5		Object ID of a String Variable object that contains the label string (32 characters maximum) or $FFFF_{16}$ if no text is supplied
Byte	6		Font type (see Annex K) (ignored if String Variable object reference is NULL or the string contains a WideString (see 4.6.19.7).
Bytes	7, 8		Object ID of an object to be used as a graphic representation of the object label or $FFFF_{16}$ if no designator supplied. When the VT draws this object it shall be clipped to the size of a Soft Key designator (see Table A.2).

F.51 Change Object Label response

This message is sent in response to the Change Object Label command.

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	In response to Change Object Label command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 181_{10}		
	Bits	7—4 1011	Command	Command
	Bits	3—0 0101	Parameter	Change Object Label
Byte	2		Error Codes (0 = no error) Bit 0 = 1 = Invalid Object ID (not listed in Object Label Reference List object) Bit 1 = 1 = Invalid String Variable Object ID Bit 2 = 1 = Invalid font type Bit 3 = 1 = No Object Label Reference List object available in object pool Bit 4 = 1 = Designator references invalid objects Bit 5 = 1 = Any other error	
Bytes	3—8		Reserved, set to FF_{16}	

F.52 Change Polygon Point command

This command is used by a Working Set to modify a point in an Output Polygon object.

NOTE 1 This message is available in VT version 4 and later.

NOTE 2 To avoid repetitive polygon draws in the case where several points need to be changed, Working Sets could use the Lock/Unlock Mask command before changing the points.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 182_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	0110	Parameter	Change Polygon Point
Bytes	2, 3			Object ID of the Output Polygon object to change	
Byte	4			Point index of the point to replace. NOTE The first point in the polygon point list is at index zero (0).	
Bytes	5, 6			New X value of a point relative to the top left corner of the polygon	
Bytes	7, 8			New Y value of a point relative to the top left corner of the polygon	

F.53 Change Polygon Point response

The VT uses this message to respond to the Change Polygon Point command.

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	In response to Change Polygon Point command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 182_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	0110	Parameter	Change Polygon Point
Bytes	2, 3			Object ID of the Output Polygon object to change	
Byte	4			Error Codes (0 = no error) Bit 0 = 1 = Invalid Object ID Bit 1 = 1 = Invalid point index Bit 2 = 1 = Any other error	
Bytes	5—8			Reserved, set to FF_{16}	

F.54 Change Polygon Scale command

This command is used by a Working Set to change the scale of a complete Output Polygon object. This message causes the value of the polygon points to be changed. For consistent implementation, the following algorithm shall be used to calculate the new points.

It is similar to the Change Size command except that is also causes the VT to rescale the polygon points. When the VT receives this message, it shall change the enclosing area of the polygon (i.e. width and height attributes) and shall adjust all polygon point positions using the following 32 bit unsigned integer algorithm:

Using unsigned 32 bit integer math:

$$new\_x = [(old\_x \times new\_width) + (old\_width/2)]/old\_width$$

$$new\_y = [(old\_y \times new\_height) + (old\_height/2)]/old\_height$$

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 183_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	0111	Parameter	Change Polygon Scale
Bytes	2, 3			Object ID of a Output Polygon object to scale	
Bytes	4, 5			New width attribute	
Bytes	6, 7			New height attribute	
Byte	8			Reserved, set to FF_{16}	

F.55 Change Polygon Scale response

The VT uses this message to respond to the Change Polygon Scale command.

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	In response to Change Polygon Scale command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 183_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	0111	Parameter	Change Polygon Scale
Bytes	2, 3			Object ID of Output Polygon object	

Bytes	4, 5	New width attribute
Bytes	6, 7	New height attribute
Byte	8	Error Codes (0 = no error) Bit 0 = 1 = Invalid object id Bits 1-3 = Undefined, set to 0 recommended Bit 4 = 1 = Any other error

F.56 Graphics Context command

This command is used to manipulate a graphics Context object (only on version 4 or later VTs). For messages larger than 8 bytes, Transport Protocol is used. Commands smaller than 8 bytes shall be padded to 8 bytes with FF_{16}. The graphics drawn by this command shall be clipped to the size of the canvas. If drawing commands place the graphics cursor outside the defined area of the object, the VT shall clip the drawing to the defined edges of the object but shall move the graphics cursor to the new end position outside the bounds of the object. The drawing rules for these graphics commands are the same as the drawing rules for normal VT Objects as specified in B.10 (for example always using a "square" brush).

For drawing, the foreground colour specified is either the foreground colour attribute of the Graphics Context Object, or the Line Colour specified in the Line Attributes object. Which one is used is determined by the state of Options bit 1.

For drawing, the background colour specified is either the background colour attribute of the Graphics Context Object, or the Fill Colour specified in the Fill Attributes object. Which one is used is determined by the state of Options bit 1.

For drawing text, the foreground colour specified is either the foreground colour attribute of the Graphics Context Object, or the Font colour specified in the Font Attributes object. Which one is used is determined by the state of Options bit 1.

For zooming, a zoom value of 1,0 means no magnification or a 1:1 mapping of pixels of the viewport to the canvas. A zoom value of 2,0 means 2:1 magnification (or zoom in), 3,0 means 3:1 magnification etc. A zoom value of 0,5 means 1:2 demagnification (or zoom out), 0,25 means 1:4 demagnification etc.

When zooming in, for example, a zoom value of 3,0 for 3:1 magnification, each pixel of the canvas is displayed in the viewport as 3 pixels wide and 3 pixels high.

When zooming out, for example, a zoom value of 0,25 for 1:4 demagnification, each block of 4 pixels wide by 4 pixels high of the canvas are displayed in the viewport as a single pixel. The exact zooming algorithm is VT proprietary. A closest colour match can be used by the VT when merging or stretching pixels.

Only the viewport is zoomed — not the canvas. The Viewport X and Viewport Y positions are in reference to the unzoomed canvas. This means that the zoom anchor point is the upper left corner of the viewport. Zooming without moving the Viewport X and Viewport Y positions, gives the appearance of stretching the image from the top left towards the bottom right.

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	On request
Data length:	Variable
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 184_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	1000	Parameter	Graphics Context Command
Bytes	2, 3			Object ID of a Graphics Context object	
Byte	4			Sub-Command ID	
Bytes	5—n			Parameters based on sub-command ID byte	

Table F.1 — Graphic command summary

Sub-Command ID	Description	Parameters	Parameter range
0	Set Graphics Cursor: This command alters the graphics cursor X/Y attributes of the object.	Bytes 5—6 = X position	−32768 to +32767
		Bytes 7—8 = Y position	−32768 to +32767
1	Move Graphics Cursor: This command alters the graphics cursor X/Y attributes of the object by moving it relative to its current position.	Bytes 5—6 = X offset	−32768 to +32767
		Bytes 7—8 = Y offset	−32768 to +32767
2	Set Foreground Colour: This command modifies the foreground colour attribute. The graphics cursor is not moved.	Byte 5 = Foreground colour	0 to 255 depending on VT's colour depth
3	Set Background Colour: This command modifies the background colour attribute. The graphics cursor is not moved.	Byte 5 = Background colour	0 to 255 depending on VT's colour depth
4	Set Line Attributes Object ID: This command modifies the Output Line object attribute. All drawing commands that follow use the new attribute value. For line suppression, set the Object ID to NULL. The graphics cursor is not moved.	Bytes 5—6 = Object ID of a Line Attributes Object or NULL for line suppression.	0 to 65534, 65535
5	Set Fill Attributes Object ID: This command modifies the fill object attribute. All drawing commands that follow use the new attribute value. For no filling, set the Object ID to NULL. The graphics cursor is not moved.	Bytes 5—6 = Object ID of a Fill Attributes Object or NULL for no further filling	0 to 65534, 65535
6	Set Font Attributes Object ID: This command modifies the font object attribute. All drawing commands that follow use the new attribute value. If text is not being used, the object can be set to NULL. The graphics cursor is not moved.	Bytes 5—6 = Object ID of a Font Attributes Object or NULL for no Font Attributes	0 to 65534, 65535

Table F.1 *(continued)*

Sub-Command ID	Description	Parameters	Parameter range
7	Erase Rectangle: Fills the rectangle at the graphics cursor using the current background colour. For this command, the Fill Attributes Object is not used regardless of the state of Options bit 1 The graphics cursor is moved to the bottom right pixel inside of the rectangle.	Bytes 5—6 = Width	0 to 65535
		Bytes 7—8 = Height	0 to 65535
8	Draw Point: Sets the pixel to the foreground colour. The graphics cursor is moved to the defined point.	Bytes 5—6 = X offset of pixel relative to the Graphics Cursor X	−32768 to +32767
		Bytes 7—8 = Y offset of pixel relative to the Graphics Cursor Y	−32768 to +32767
9	Draw Line: Draws a line from the graphics cursor to the specified end pixel using the foreground colour. The Output Line Object drawing rules apply with respect to the end pixel location and Line Attributes. The graphics cursor is moved to the specified end pixel.	Bytes 5—6 = X offset of end pixel relative to the Graphics Cursor X	−32768 to +32767
		Bytes 7—8 = Y offset of end pixel relative to the Graphics Cursor Y	−32768 to +32767
10	Draw Rectangle: Draws a rectangle at the graphics cursor. The Rectangle Object drawing rules apply. If a Line Attributes object is currently defined, the border is drawn. If a fill attribute object is currently defined, the rectangle is filled. The graphics cursor is moved to the bottom right pixel inside of the rectangle.	Bytes 5—6 = Width	0 to 65535
		Bytes 7—8 = Height	0 to 65535
11	Draw Closed Ellipse: Draws a closed ellipse bounded by the rectangle defined by the current graphics cursor location and the width and height given. The Output Ellipse object drawing rules apply. If a Line Attributes object is currently defined, the border is drawn. If a fill attribute object is currently defined, the ellipse is filled. The graphics cursor is moved to the bottom right pixel inside of the bounding rectangle.	Bytes 5—6 = Width	0 to 65535
		Bytes 7—8 = Height	0 to 65535
12	Draw Polygon: Draws a polygon from the graphics cursor to the first point, then to the second point and so on. The polygon is closed if the last point has the offset 0,0. This is because offset 0,0 gives the coordinates of the original graphics cursor which was used as the first point in the polygon. If the data does not close the polygon, no automatic closing is performed and filling is ignored. Foreground colour is used for the border colour. The Output Polygon object drawing rules apply. If a Line Attributes object is currently defined, the border is drawn. If a fill object is currently defined and the polygon is closed, the polygon is filled. The graphics cursor is moved to the last point in the list.	Byte 5 = Number of polygon points to follow	0 to 255
		Bytes 6—7 = First point offset X value relative to the Graphics Cursor X (signed)	−32768 to +32767
		Bytes 8—9 = First point offset Y value relative to the Graphics Cursor Y (signed) ... { list of points continues starting at byte 10 with each point requiring 4 bytes of data}	−32768 to +32767

Table F.1 *(continued)*

Sub-Command ID	Description	Parameters	Parameter range
13	Draw Text: Draws the given text using the Font Attributes object. Any flashing bits in the Font style of the Font Attributes object are ignored If opaque, the background colour attribute is used. The graphics cursor is moved to the bottom right corner of the extent of the text.	Byte 5: 0 = Opaque, 1 = Transparent.	0 or 1
		Byte 6 = Number of bytes to follow	0 to 255
		Bytes 7 — *n* = Text string. The text can be either 8-bit or WideString (see 4.6.19.7).	
14	Pan Viewport: This command modifies the viewport X and Y attributes and forces a redraw of the object. This allows “panning” of the underlying object contents. The graphics cursor is not moved.	Bytes 5—6 = Viewport X attribute	−32768 to +32767
		Bytes 7—8 = Viewport Y attribute	−32768 to +32767
15	Zoom Viewport: This command allows magnification of the viewport contents. See section on zooming for meaning of the zoom value. The graphics cursor is not moved.	Byte 5—8 = “zoom” value (Float numeric).	−32.0 to +32.0
16	Pan and Zoom Viewport: This command allows both panning and zooming at the same time combining commands 14 and 15.	Bytes 5—6 = Viewport X attribute	−32768 to +32767
		Bytes 7—8 = Viewport Y attribute	−32768 to +32767
		Byte 9—12 = “zoom” value (Float numeric).	−32.0 to +32.0
17	Change Viewport Size: This command changes the size of the viewport and can be compared to the normal Change Size command. The graphics cursor is not moved. NOTE The size of the object (i.e. the memory used) cannot be changed.	Bytes 5—6 = New width	0 to 32767
		Bytes 7—8 = New height	0 to 32767

Table F.1 *(continued)*

Sub-Command ID	Description	Parameters	Parameter range
18	Draw VT Object: This command draws the VT Object specified by the Object ID in bytes 5—6 at the current graphics cursor location (top left corner). Any drawable object can be specified with the exception of the Graphics Context object specified in bytes 2—3 or any object that contains this Graphics Context object (circular references are not allowed). The object shall be drawn using the current value and state of that object at the time the command was specified (for instance, enabled or disabled), except flashing bitmaps which are drawn regardless of their flashing state. A focus indicator, however, shall not be drawn even if the specified object (or any child object) has focus at that time. Also, if the object is being edited by the operator, it shall be drawn as if it is not being edited, using the last accepted value of the object (not a temporary value that the operator is still entering). The graphics cursor is moved to the bottom right corner of the object that was drawn. Normal VT Object transparency rules apply when drawing the VT Object onto the canvas. Any colours outside of the colours allowed by this Graphics Context Object (Table B.41) shall be treated as transparent.	Bytes 5—6 = Object ID of object to draw	0 to 65534
19	Copy Canvas to Picture Graphic: This command copies the current canvas of the Graphics Context Object into the Picture Graphic Object specified by the Object ID in bytes 5—6. If the Picture Graphic is smaller than the canvas, then it shall be clipped to fit within the Picture Graphic. If the Picture Graphic is larger than the canvas, then the extra pixels in the Picture Graphic are not changed. Colours in the Canvas that are set to the transparency colour in the Graphics Context Object are not copied and the corresponding pixels in the Picture Graphic are not changed. The picture graphic shall have at least the same number of colours as the Graphics Context Object. Any colours outside of the colours allowed by this Picture Graphic Object (Table B.41) shall be treated as transparent.	Bytes 5—6 = Object ID of Picture Graphic object to copy to	0 to 65534

Table F.1 *(continued)*

Sub-Command ID	Description	Parameters	Parameter range
20	Copy Viewport to Picture Graphic: This command copies the current Viewport (zoomed or panned) of the Graphics Context Object into the Picture Graphic Object specified by the Object ID in bytes 5—6. If the Picture Graphic is smaller than the Viewport, then it shall be clipped to fit within the Picture Graphic. If the Picture Graphic is larger than the Viewport, then the extra pixels in the Picture Graphic are not changed. Colours in the Viewport that are set to the transparency colour in the Graphics Context Object are not copied and the corresponding pixels in the Picture Graphic are not changed. The picture graphic shall have at least the same number of colours as the Graphics Context Object. Any colours outside of the colours allowed by this Picture Graphic Object (Table B.41) shall be treated as transparent.	Bytes 5—6 = Object ID of Picture Graphic object to copy to	0 to 65534

F.57 Graphics Context response

The VT uses this message to respond to the Graphics Context command.

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 184_{10}		
	Bits	7—4 1011	Command	Command
	Bits	3—0 1000	Parameter	Graphics Context Command
Bytes	2, 3		Object ID of a Graphics Context object	
Byte	4		Sub-command ID	
Byte	5		Error codes (0 = no errors) Bit 0 = 1 = Invalid Object ID or object is not a Graphics Context object Bit 1 = 1 = Invalid sub-command id Bit 2 = 1 = Invalid parameter Bit 3 = 1 = Sub command will produce invalid results Bit 4 = 1 = Any other error	
Bytes	6—8		Reserved, set to FF_{16}	

F.58 Get Attribute Value message

This command is used by a Working Set to query the VT for the current state of objects within the VT.

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 185_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	1001	Parameter	Get Attribute Value
Bytes	2, 3			Object ID	
Byte	4			Attribute ID of the Object	
Bytes	5—8			Reserved, set to FF_{16}	

F.59 Get Attribute Value response

The VT uses this message to respond to a Get Attribute Value message.

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	In response to Get Attribute Value message
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 185_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	1001	Parameter	Get Attribute Value Response
Bytes	2, 3			Object ID or $FFFF_{16}$ to indicate an error response	
Byte	4			Attribute ID of the Object	
No Error Response:					
Bytes	5—8			Current value of the attribute. Size depends on attribute data type. Values greater than 1 byte are transmitted little endian (LSB first):	

Boolean:	1 byte for TRUE/FALSE
Integer:	1, 2 or 4 bytes as defined in object tables
Float:	4 bytes
Bitmask:	1 byte

Error Response:

Bytes	5, 6	Object ID
Byte	7	Error Codes When bytes 2,3 = $FFFF_{16}$ (the error response), one or more of the bits below shall be set Bit 0 = 1 = Invalid Object ID Bit 1 = 1 = Invalid Attribute ID Bits 2, 3 = Reserved, set to 0 Bit 4 = 1 = Any other error
Byte	8	Reserved, set to FF_{16}

F.60 Select Colour Map or Palette command

The Select Colour Map or Palette command is used to alter the colours of the presentation. The command applies to any presentation from the originating Working Set, which includes objects that can be shown on other Working Set screens (e.g. Auxiliary Control objects as can be presented on VT proprietary and other Working Set masks using the Auxiliary Control Designator Type 2 Object Pointer).

For VT version 4 and later, the Object ID referenced by this command can be a Colour Map object. For VT version 6 and later, the Object ID referenced can alternately be a Colour Palette object.

When this command is processed by a VT that is compatible with VT version 6 and later, the Working Set Special Controls object (see B.29) shall be updated according to the referenced object.

NOTE 1 This command can take a long time to execute.

NOTE 2 This message is available in VT version 4 and later.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 186_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	1010	Parameter	Select Colour Map or Palette
Bytes	2, 3			Object ID of the Colour Map object, or the Colour Palette object, or $FFFF_{16}$ to restore the default colour mapping (see A.3).	
Bytes	4—8			Reserved, set to FF_{16}	

F.61 Select Colour Map or Palette response

NOTE This message is available in VT version 4 and later.

Transmission repetition rate:	In response to Select Colour Map or Palette command
Data length:	8 bytes
Parameter group number:	VT to ECU, Destination-Specific
Allowed in a Macro:	No

Byte	1	VT function = 186_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	1010	Parameter	Select Colour Map or Palette
Bytes	2, 3			Object ID of the Colour Map object	
Byte	4			Error Codes (0 = no error) Bit 0 = 1 = Invalid object id Bit 1 = 1 = Invalid Colour Map or Palette Bit 2 = 1 = Any other error	
Bytes	5—8			Reserved, set to FF_{16}	

F.62 Execute Extended Macro command

This command is used to execute a Macro.

NOTE This message is available in VT version 5 and later.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a Macro:	Yes

Byte	1	VT function = 188_{10}			
	Bits	7—4	1011	Command	Command
	Bits	3—0	1100	Parameter	Execute Extended Macro
Bytes	2, 3			Object ID of Macro object	
Bytes	4—8			Reserved, set to FF_{16}	

F.63 Execute Extended Macro response

The VT uses this message to respond to the Execute Extended Macro command.

NOTE This message is available in VT version 5 and later.