INTERNATIONAL STANDARD

ISO 11783-6

Second edition
2010-10-15

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 11783-6:2010(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

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.

ISO 11783-6 was prepared by Technical Committee ISO/TC 23, *Tractors and machinery for agriculture and forestry*, Subcommittee SC 19, *Agricultural electronics*.

This second edition cancels and replaces the first edition (ISO 11783-6:2004), which has been technically revised. It also incorporates the Technical Corrigendum ISO 11783-6:2004/Cor.1:2005. It introduces requirements for Version 4 VTs and Working Sets, while retaining the requirements of the first edition for Version 3 VTs and Working Sets. The specific requirements of Annex G, as well as Annex J and the revised Table L.2, are also applicable to Version 3.

ISO 11783 consists of the following parts, under the general title *Tractors and machinery for agriculture and forestry — Serial control and communications data network*:

— *Part 1: General standard for mobile data communication*

— *Part 2: Physical layer*

— *Part 3: Data link layer*

— *Part 4: Network layer*

— *Part 5: Network management*

— *Part 6: Virtual terminal*

— *Part 7: Implement messages application layer*

— *Part 8: Power train messages*

— *Part 9: Tractor ECU*

— *Part 10: Task controller and management information system data interchange*

— *Part 11: Mobile data element dictionary*

— *Part 12: Diagnostics services*

— *Part 13: File server*

— *Part 14: Sequence control*

Introduction

Parts 1 to 14 of ISO 11783 specify a communications system for agricultural equipment based on the CAN 2.0 B[1] protocol. SAE J 1939[2] 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 J 1939 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[3]. 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 part of ISO 11783 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. Linear Bar Graph object, Change Numeric Value command). This aids in the recognition of each of these terms as being a specific item having a specific definition in the document.

The International Organization for Standardization (ISO) draws attention to the fact that it is claimed that compliance with this part of ISO 11783 may involve the use of a patent concerning the controller area network (CAN) protocol referred to throughout the document.

ISO takes no position concerning the evidence, validity and scope of this patent.

The holder of this patent has assured ISO that he is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with ISO. Information may be obtained from:

Robert Bosch GmbH
Wernerstrasse 51
Postfach 30 02 20
D-70442 Stuttgart-Feuerbach
Germany

Attention is drawn to the possibility that some of the elements of this part of ISO 11783 may be the subject of patent rights other than that those identified above. ISO shall not be held responsible for identifying any or all such patent rights.

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 and mounted, semi-mounted, towed or self-propelled implements. Its purpose is to standardize the method and format of transfer of data between sensors, actuators, control elements and information storage and display units, whether mounted on, or part of, the tractor or implement. This part of ISO 11783 describes a universal virtual terminal (VT) that can be used by both tractors and implements.

It is applicable to both Version 3 and Version 4 VTs and Working Sets.

2 Normative references

The following referenced documents are indispensable for the application 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 11783-1, *Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 1: General standard for mobile data communication*

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, definitions and abbreviated terms

For the purposes of this document, the terms, definitions and abbreviated terms given in ISO 11783-1 and the following terms and definitions apply.

3.1
auxiliary input unit
autonomous control function (CF) providing auxiliary controls for common use that may 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 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 references a specific object within an object pool

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

NOTE 1 AID values range from 0 to FE_{16} (254_{10}), with 255 as the NULL_AID.

NOTE 2 This field appears in the attribute and record format tables. AIDs that are explicitly defined with square brackets, [ ], are read-only attributes using the Get Attribute Value message. AIDs that are explicitly defined without square brackets are writable with the Change Attribute command.

3.5
char
single character where the size is 1 byte

NOTE Commonly used for ISO 8859 characters (e.g. 41_{16} in ISO/IEC 8859-1 represents “A”), see Annex L.

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

NOTE Size is variable, based on the encoding scheme (see char and WideChar).

3.7
code plane
group of 65 536 possible character codes

NOTE Unicode/ISO 10646 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 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 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 UTF-16 encoding of characters in code plane 1 to 16 (see 4.6.16.6).

NOTE 2 The UTF-16 character encoding scheme is 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”.

NOTE 1 See Annex L.

NOTE 2 Two WideChars can be combined to indicate character codes exceeding 16 bits (see 4.6.16.6).

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 String length is variable.

3.14
VT number
identification number that is derived from the function instance of the VT

NOTE VTs can then be referenced as VT Number 1, VT Number 2, etc.

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

NOTE See 4.1 and 4.7.

3.16
window cell
equal-sized cell in a grid on a User-Layout Data Mask

NOTE See 4.7.

3.17
window mask object
supplied by the Working Set for placement by the operator into the area of one or more window cells but not a partial cell

NOTE 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 See 4.7.

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

NOTE See 4.7.

3.20
Key Group Object
area of one or more Key Cells and containing a grouping of one or more Key Objects

NOTE See 4.7.

3.21
non-VT screen
VT-capable display on which no Data Masks are visible

NOTE See 4.7.

3.22
non-VT area
area outside the normal Data Mask and Soft Key Mask visible at the same time as a Data Mask and Soft Key Mask are displayed

NOTE See 4.7.

3.23
range
value
signifying that each object has an attributes and record format table

NOTE Each parameter has a range or value for the specific attribute. Where there are ranges and values that are explicitly defined with square brackets, [ ], they are applicable to VT Version 3 and prior, whereas the adjacent unbracketed range or value applies to VT Version 4 or later.

4 Technical requirements

4.1 Overview

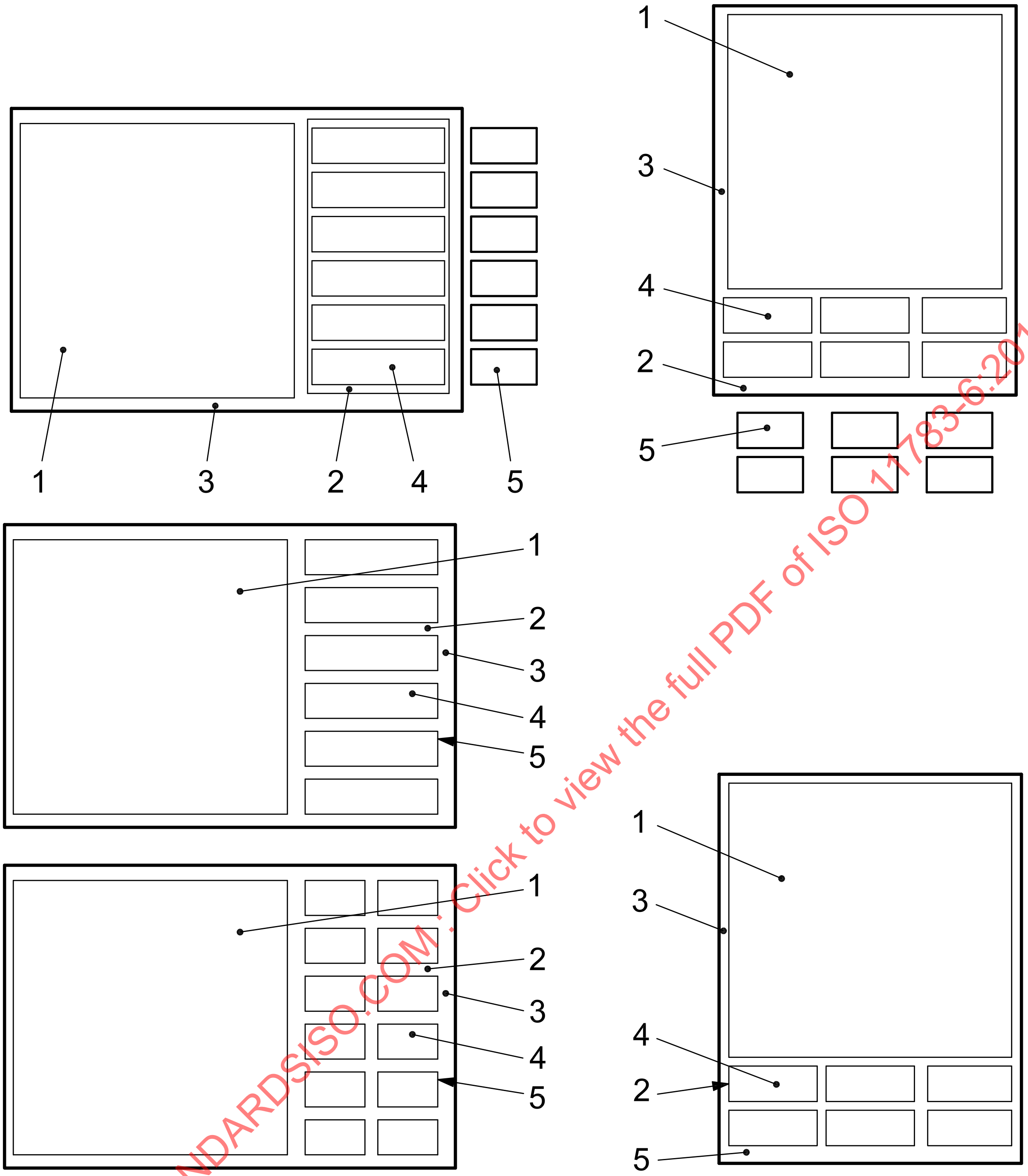
A virtual terminal (VT) is a control function (CF) within an electronic control unit (ECU), consisting of a graphical display and input functions, connected to an ISO 11783 network that provides the capability for a CF, composing an implement or a group of implements to interact with an operator. The 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 part of ISO 11783, the term *Working Set* will be used for a CF, as an implement or a group of implements represented either 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 and control keys of the VT to allow the operator to send signals back to the Working Set.

This part of ISO 11783 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 the VT services. The interface protocol of this part of ISO 11783 also reduces the runtime ISO 11783 communication bus traffic as much as possible. For these reasons, the requirements of this part of ISO 11783 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 manufacturers' 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.



Key

1	Data Mask Area
2	Soft Key Mask area
3	physical screen
4	Soft Key designator
5	physical Soft Key

Figure 1 — Virtual terminal — Examples

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**

This 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**

This 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**

This 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. Data input operation that originates on a User-Layout Data Mask does not affect the VT Status message.

There are two types of Data Input: “editing” and “real time editing”.

1) **Editing**

This 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 ESC means either may be a permanent key or may only be available during data entry (see Table 4). 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 input**

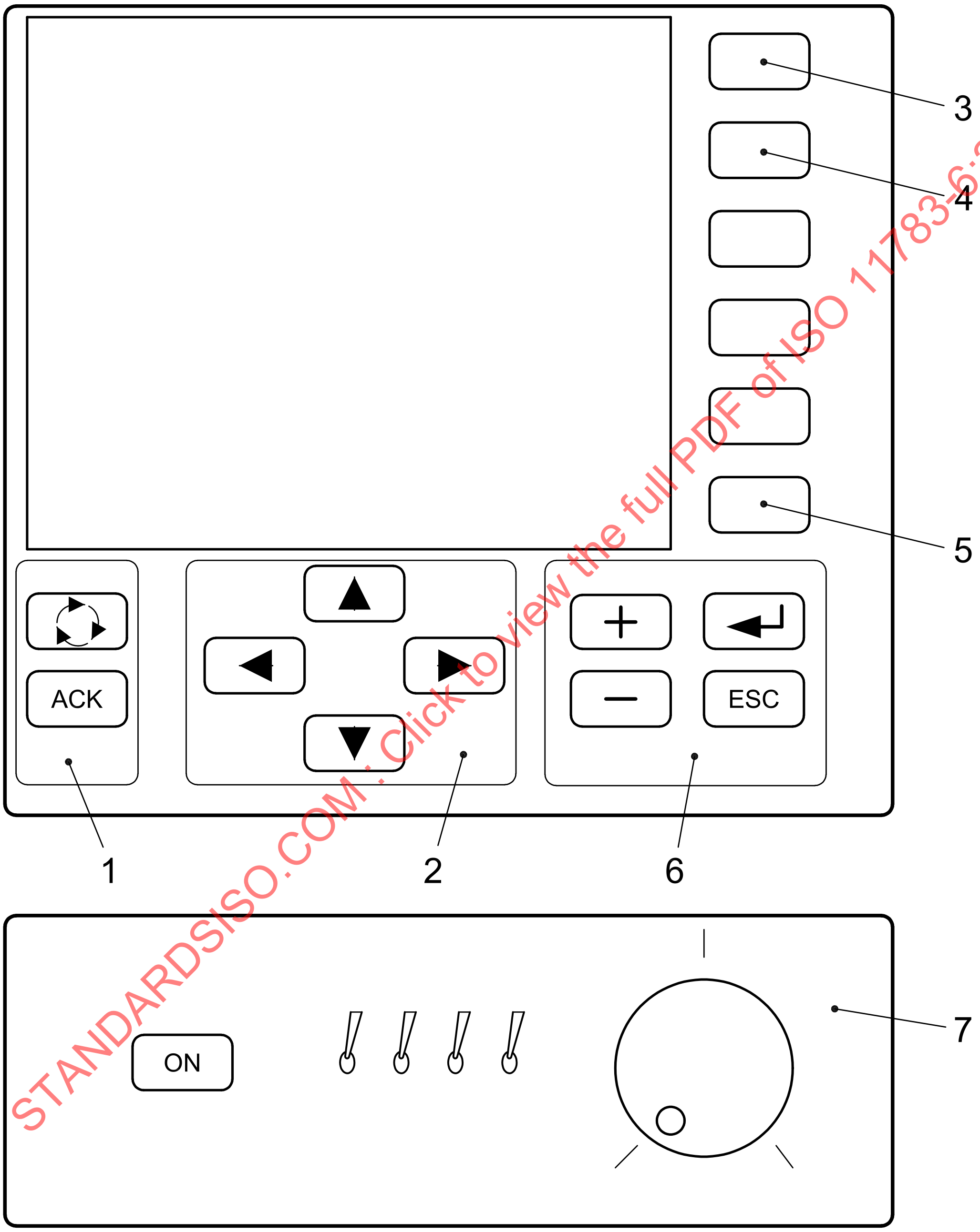
This is a means of data input for an Input Number object and Input List object where the object has focus and is open for operator input, and changes by the operator to the value are periodically transmitted to the Working Set while the object is being changed. The VT Change Numeric Value message is limited to a 5 Hz update rate. Each value change sent to the Working Set is considered a complete transaction 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 data input, the VT shall ensure that the on-screen value is equal to the value last sent to the Working Set. The VT may 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 data input shall meet the operator controls requirement specified in ISO 15077.

d) **Control**

This is a means of selecting between Working Sets whenever a Data Mask is visible, and a means of 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**

This 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 may be a simple on/off type buzzer or an acoustic component capable of variable frequency and audio level.

4.4 Coordinate system

Positions and sizes in this part of ISO 11783 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 corner of the parent object.

4.5 Display areas

The following defines standard Data Mask and Soft Key Mask areas of the display. Alternative usage of these areas supports displaying data from multiple working sets (see 4.7).

4.5.1 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. This requirement does not limit the physical resolution or size of the display, only the usable 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, and
- 480 × 480.

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.

4.5.2 Soft Key Mask area and Soft Key designators

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. The Soft Key designators may contain text, graphics or both. The Soft Key Mask area may 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). It is recommended that this visible separation be drawn outside the Soft Key designator area.

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 six Physical Soft Keys.

c) VT Version 3 and prior shall support a maximum of 64 virtual Soft Keys per Soft Key Mask (see 4.5.2.2).

d) VT Version 4 and later shall support exactly 64 virtual Soft Keys per Soft Key Mask (see 4.5.2.2).

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 Soft Keys using the six physical keys.

4.5.2.1 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 must provide physical buttons for the Soft Keys. For example, on a VT with touch screen, the physical Soft Keys may 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.

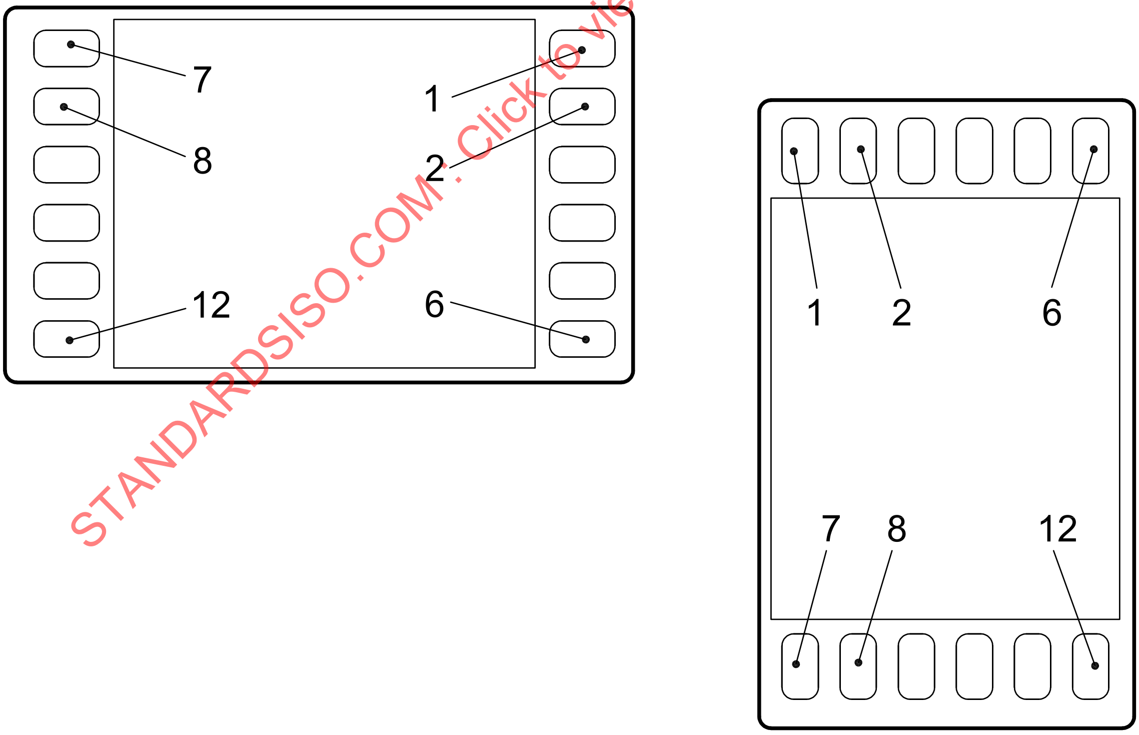


Figure 3 — Physical Soft Key orientation examples showing key locations

4.5.2.2 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 Set's Soft Keys.

4.5.2.3 Navigation Soft Keys

Navigation Soft Keys is the count of the number of physical Soft Keys that the VT may 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 another means of navigation that does not use the physical Soft Keys, this value shall be zero.

4.5.2.4 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 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 may 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) may not completely fill the Soft Key Mask. The remainder of the Soft Key designators shall not be used.

The VT shall not provide navigation to any trailing Soft Keys while those items are Pointers to NULL Object ID.

As shown in Figure 4, a VT is designed with six physical Soft Keys, 64 virtual Soft Keys, and one [a) in Figure 4] navigation Soft Key. The Working Set provides 18 Soft Keys to the VT; however, there are three which are Pointers to 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 five Soft Keys [a) in Figure 4], starting with the first Soft Key. When the navigation key is pressed, the VT shows the next group of Soft Keys. Another example [b) in Figure 4] shows a similar example with two navigation Soft Keys. Another example [c) in Figure 4] 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 [d) in figure].

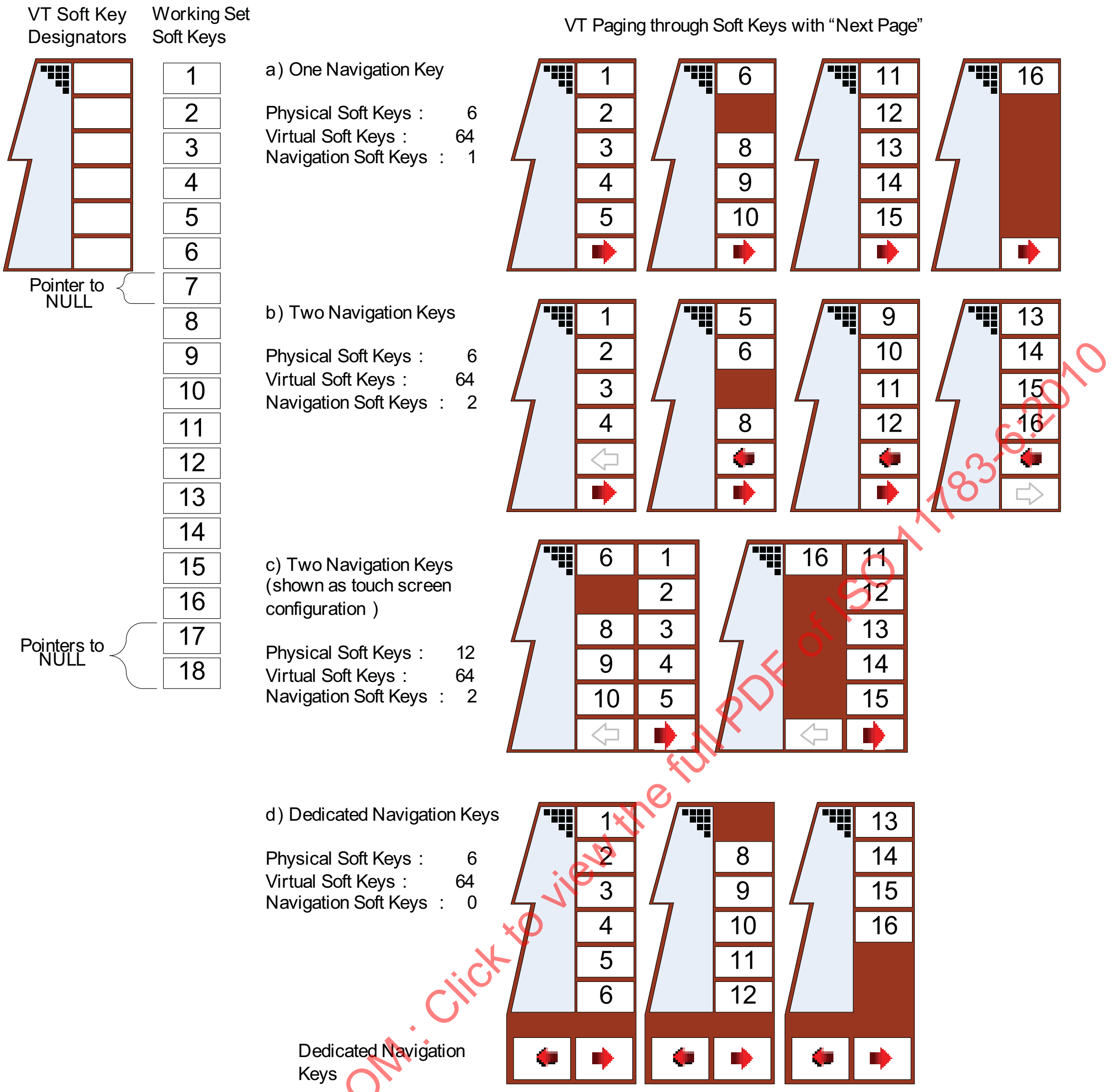


Figure 4 — VT virtual Soft Key paging

4.6 Behaviour

4.6.1 Object pools

4.6.1.1 General

The operator interface definition for a device or 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*. These objects are defined in detail in the following subclauses. 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 may not be between different Working Sets.

The object pool is transferred to the VT at initialization by using the transport protocol described in ISO 11783-3, and the extended transport protocol specified in Annex K. The procedure is described in more detail in Annex C. The VT is intended to be capable of storing the object pools in a modifiable memory area. The size and number of object pools are limited only by the VT's available memory and software design, but only one object pool per Working Set exists. All objects shall be fully described before they are made active in a mask on the display.

4.6.1.2 NULL Object ID

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

4.6.1.3 Processing objects

Objects listed in parent objects may 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.

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 ISO 11783-5 NAME 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.

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 runtime.

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 is never considered to be a Working Set and therefore 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 respect of responses being directed only to the Working Set Master. See Table 1.

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 1 to 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.

In configuration 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 Widechar 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.

4.6.3 Language, formats and measurement units selection

The following is applicable to VTs.

- The VT shall send the standard language, format and measurement units messages defined in ISO 11783-7, hereafter referred to as “standard setups”. The Working Set object identifies the languages that the Working Set supports. The VT shall provide a method for the operator to view the list of supported languages and to select an item from the list. If no language has been entered by the operator (as would be the case in a factory-new VT), the VT shall attempt to query the default language from the tractor ECU. Once the operator has set the language, the VT's language message always takes priority over the tractor ECU's language.

- The VT shall also provide a method for the operator to select formats (Time, Date, etc.) and measurement units. The VT shall report 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 — and to the selected units — e.g. by changing offsets and scales. If the Working Set does not support the specified language, formats or units, it shall use a proprietary method to select an appropriate setting, and can therefore choose default units as specified by the Working Set designers.

- The VT shall store the standard setups in non-volatile storage and restore the values during initialization.

- The VT shall respond to ISO 11783-7 “Language Command” requests sent to the global address (GA).

- The VT shall respond to ISO 11783-7 “Language Command” requests directed to the VT.

The following is applicable to Working Sets.

- Working sets 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).

- Working Sets shall use their default language if the selected language is not supported.

4.6.4 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.

a) 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 at least 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 shall attempt to request the default language setting from the tractor ECU.

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

b) 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 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.

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

   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 may use this to add or remove members from the set. No Working Set initialization is required.

5) The Working Set Master shall transmit the Working Set Maintenance message once with the "initiating flag" set to 1.

   If the VT had previously detected a shutdown and was transmitting the NACK in response to the Working Set Maintenance message (see 4.6.6), there are two cases where the VT shall stop transmitting the NACK:

   i) In the case of a Version 3 or later ECU, identified by Working Set Maintenance message; Byte 3 $<$ 255 and the initiating flag Byte 2 Bit 0 $=$ 1.

   ii) In the case of a version 2 and prior ECUs, identified by Working Set Maintenance message; Byte 2 $=$ FF_{16}, and the VT received a Working Set Master message since the prior maintenance message.

6) The Working Set Master shall begin transmitting the Working Set Maintenance message with the "initiating flag" set to 0.

7) The Working Set Master may 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 a presentation that is language- or unit-specific.

8) The Working Set Master may 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.

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

10) 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 transport protocol (see ISO 11783-3), extended transport protocol (see Annex K) and the messages given in Annex C.

c) **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, and

v) 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 may initialize connection to any other VT on the network. The Working Set may provide a means for the operator to set the maximum wait time period or it may be obtained from the boot time specification in the “Get Hardware response” message of the preferred VT.

4.6.5 Working Set object and active masks

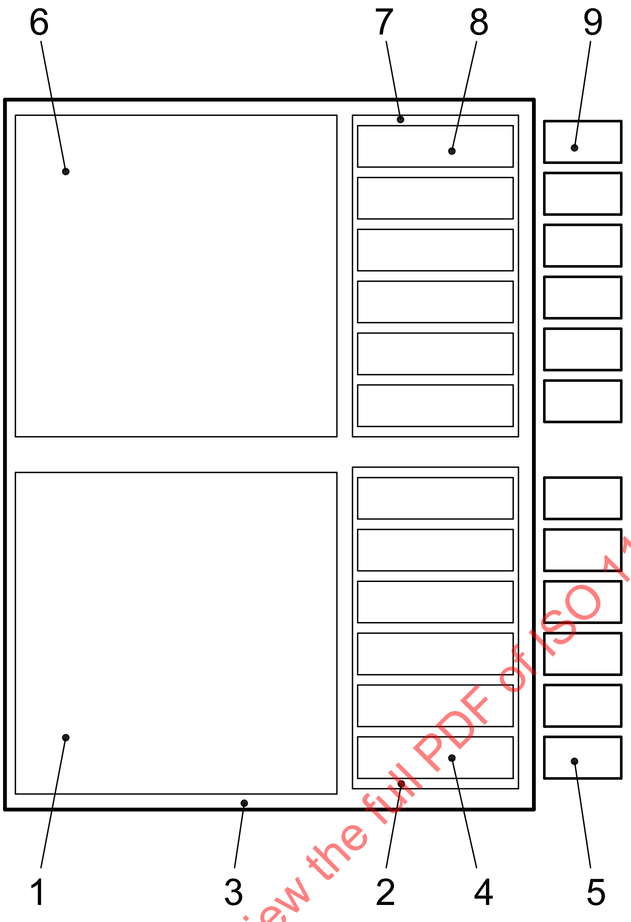
In the initial object pool definition, each Working Set Master shall provide one, and only one, Working Set object in order to define a descriptor, active mask and supported languages for the Working Set. The descriptor may be graphical, text or both, but shall fit inside the area defined by the VT for a Soft Key designator. The descriptor may 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 could 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.

For Version 4 and later, a VT may also display one or more Working Sets which are not active in addition to the Active Working set (see Figure 5). 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 NACK, or 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. If the Working Set does not support this feature, the VT shall inform the operator that the displayed information may not be updated. The VT may still display the inactive Working Set, because the inactive Working Set may update its data (see 4.6.7).

For Working Set state changes, see Table 2.



Key

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

Figure 5 — VT displaying active and inactive Working Sets simultaneously — Example

Table 2 — Working Set state changes

VT behaviour		
Working set state from/to	**VT displays only active Working Set**	**VT displays active and inactive Working Sets simultaneously**
Active to Inactive	Hide the Working Set Hide the associated Soft Key Mask Send the VT Status message to the global address (255) to inform Working Sets of the change[a]	Hide the Working Set, if it is not visible in its inactive state Hide the associated Soft Key Mask, if it is not visible in its inactive state Remove visual indication that the Working Set is the active Working Set Send the VT Status message to the global address (255) to inform Working Sets of the change[a] Send the On User-Layout Hide/Show message to the Working Set when it is now inactive but visible
Inactive to Active	Display the Working Set's currently active Data Mask or Alarm Mask Display the associated Soft Key Mask Visually indicate to the operator that the Working Set is the active Working Set Send the VT Status message to the global address (255) to inform Working Sets of the change[a]	Send the On User-Layout Hide/Show message when the Working Set was inactive but visible Display the Working Set's currently active Data Mask or Alarm Mask Display the associated Soft Key Mask Visually indicate to the operator that the Working Set is the active Working Set Send the VT Status message to the global address (255) to inform Working Sets of the change[a]
Hidden to Active	Display the Working Set's currently active Data Mask or Alarm Mask Display the associated Soft Key Mask Visually indicate to the operator that the Working Set is the active Working Set Send the VT Status message to the global address (255) to inform Working Sets of the change[a]	Send the VT Status message to the global address (255) to inform Working Sets of the change[a]
Active to Hidden	Hide the Working Set Hide the associated Soft Key Mask Send the VT Status message to the global address (255) to inform Working Sets of the change[a]	Send the VT Status message to the global address (255) to inform Working Sets of the change[a]
Inactive, but visible, to Hidden	(not applicable)	Send the On User-Layout Hide/Show message when the Working Set is inactive but visible

[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 a 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 may not be visible, but still remains as the active mask for that Working Set.

4.6.6 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, defined as the state in which all functions dependent on the VT operator interface are put into a known state that will not put the operator or machine at risk. The Working Set may re-establish connection to the VT by restarting the initialization procedure.

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 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:

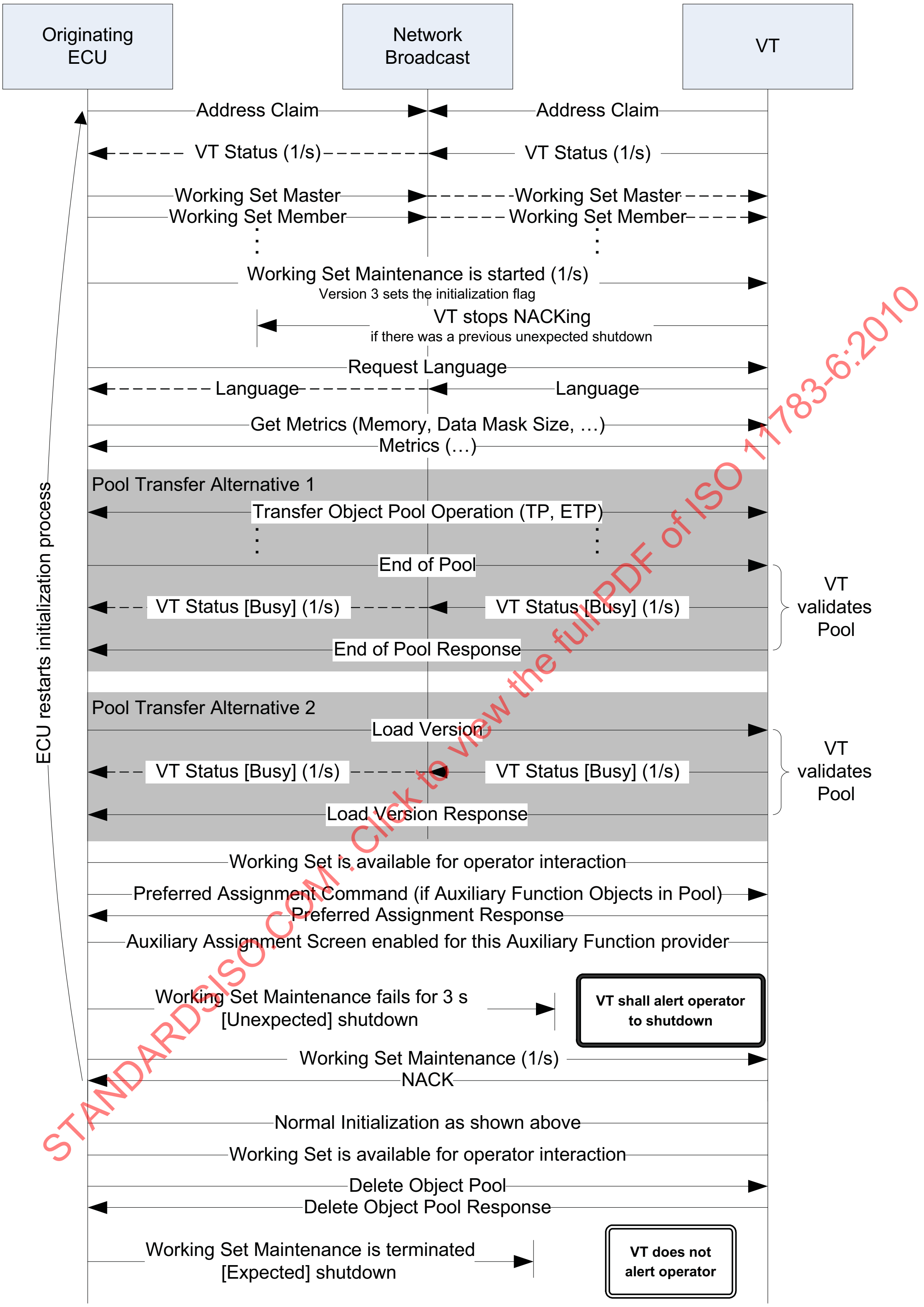
- if the Working Set has commanded the VT to delete the object pool, and the Working Set then stops sending Working Set Maintenance messages — not an unexpected shutdown situation, and which allows the Working Set to silently remove itself from the VT;

- if the VT can detect the ignition key state and the ignition key is reported as off — not considered an unexpected shutdown;

- 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 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. If the Working Set has control of the VT, the display is cleared and the VT may give control to another connected Working Set and send the VT Status message to the global address. If there is an active alarm for the failed Working Set, the VT deselects the Alarm Mask automatically.

When a Working Set's object pool has been deleted and there exist auxiliary assignments mapped to this Working Set, the VT shall remove them. When a Working Set's object pool has been deleted and the VT receives a Working Set Maintenance message from the missing Working Set, it shall NACK the message (see ISO 11783-3). The NACK message is sent to the Working Set Master's source address (see 4.6.4).


Originating ECU
Network Broadcast
VT
Address Claim
Address Claim
VT Status (1/s)
VT Status (1/s)
Working Set Master
Working Set Member
Working Set Master
Working Set Member
Working Set Maintenance is started (1/s)
Version 3 sets the initialization flag
VT stops NACKing
if there was a previous unexpected shutdown
Request Language
Language
Language
Get Metrics (Memory, Data Mask Size, …)
Metrics (…)
Pool Transfer Alternative 1
Transfer Object Pool Operation (TP, ETP)
End of Pool
VT Status [Busy] (1/s)
VT Status [Busy] (1/s)
End of Pool Response
VT validates Pool
Pool Transfer Alternative 2
Load Version
VT Status [Busy] (1/s)
VT Status [Busy] (1/s)
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 s
[Unexpected] shutdown
VT shall alert operator to shutdown
Working Set Maintenance (1/s)
NACK
ECU restarts initialization process
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

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

Figure 6 — Initialization, unexpected shutdown and expected shutdown

4.6.7 Updating the operator interface

4.6.7.1 General

CAN has finite bandwidth available in support of all services (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, 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), and

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

4.6.7.2 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. Changeable 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.

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

4.6.7.3 Changing, adding and deleting objects

Objects can be completely redefined at runtime and 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 Annex C).

The entire object pool can be deleted from the volatile memory in the VT by the Working Set sending a Delete Object Pool command.

4.6.8 Special objects

4.6.8.1 Container objects

A Container object is a special object used to

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

b) 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.

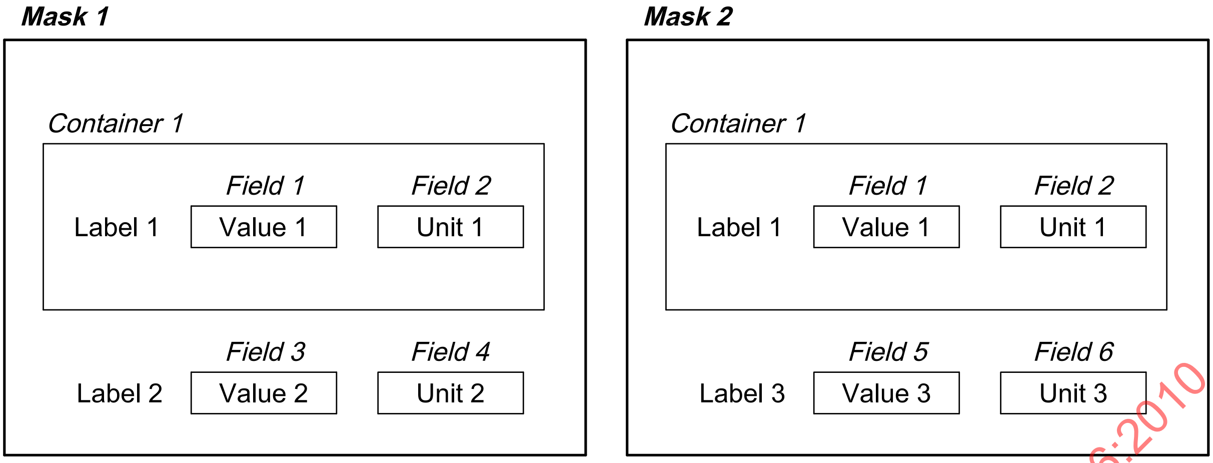
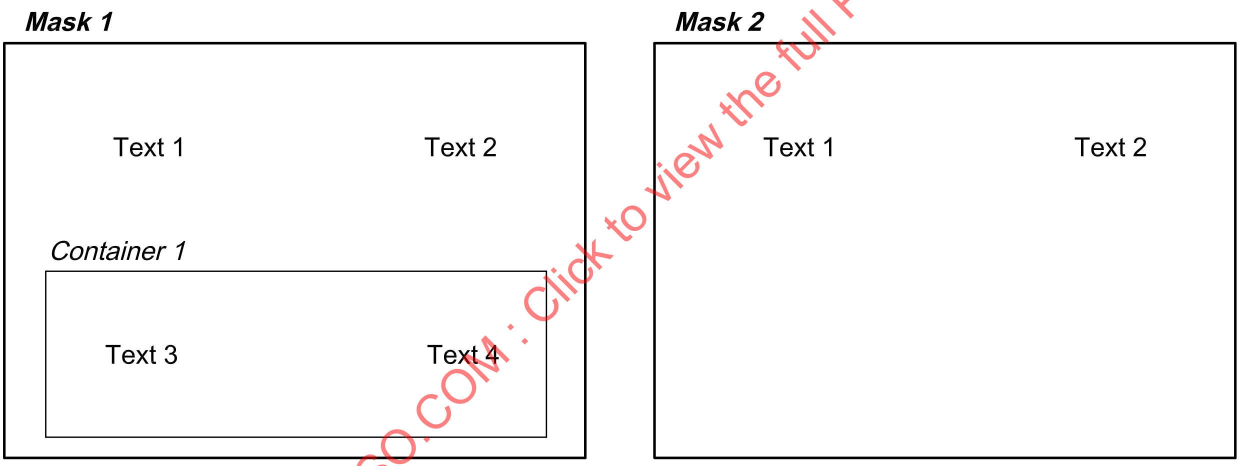


Figure 7 — Container reuse

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, whereas 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 runtime, the Working Set determines whether or not the particular feature is available. If not, the Working Set hides the container, as shown in Figure 8 b).



a) Particular feature available

b) Feature not available: container hidden

Figure 8 — Container used to hide objects — Examples

4.6.8.2 Attribute objects

There are five types of attribute object: 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.

4.6.8.3 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 the output field 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.

4.6.8.4 Macros

Macro objects are used to improve the performance of the operator interface, as follows.

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

e) Macro object IDs shall be in the range 0 to 255.

f) 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.

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 has a Macro to clear the Input String — On Input Field Deselection. The “Enter” means on an Input String of length greater than three 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 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 may end up with either value, due to the asynchronous processes.

Circular references, such as those created when a Macro triggers an event that references the same Macro, are not permitted, since they create infinite Macro loops inside the VT and could render the VT inoperable for all Working Sets.

4.6.8.5 Object pointer

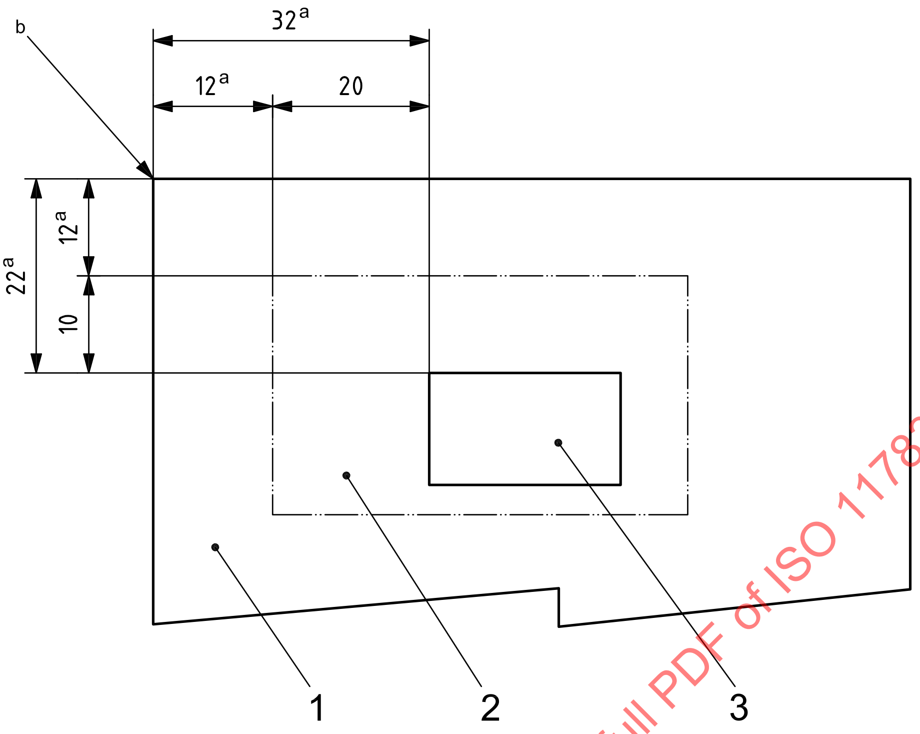
The Object Pointer object allows runtime modification of included objects. By changing the value of the pointer object, a different object can be drawn at the same location. The type of object that the Object Pointer is allowed to reference is limited and depends on the parent object.

When an Object Pointer is modified at runtime, resulting in an invalid object reference, the VT is not required to detect this object pool error immediately but may 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 “Change Active Mask response” (see F.35) or “VT Change Active Mask” message (see H.14) to inform the Working Set, and may delete the object pool.

4.6.9 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 9 shows an example Data Mask with the relative locations of several objects.

Dimensions in pixels



Key

1 Data Mask

2 container

3 input field

[a] Absolute.

[b] Point of origin, 0, 0, absolute.

Figure 9 — Relative and absolute location of objects

4.6.10 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. In this case, the objects shall by 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. 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.

When an object is changed or hidden, the VT shall update the display accordingly if the object is currently visible.

EXAMPLE Figure 10 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 10 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 10 c). The VT may implement this refresh in a manner where the intermediate display is never seen by the operator.

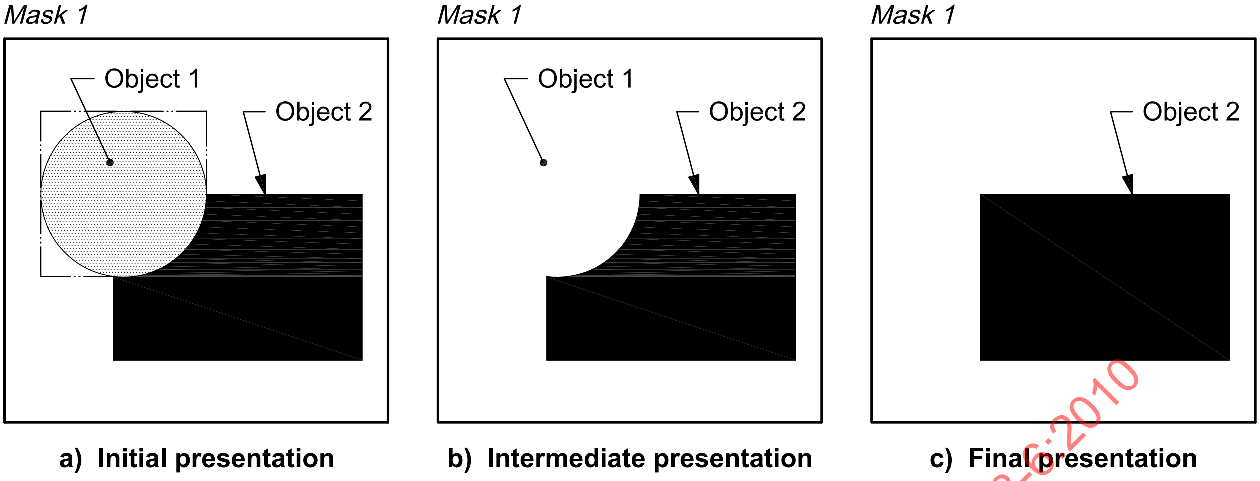


Figure 10 — Object changed or hidden — Display update

4.6.11 Alarm handling

Alarms allow a Working Set to display alarm information in the Data Mask area of the VT at any time. The Alarm Mask covers the entire Data Mask area. 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 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 asserted an Alarm Mask at the same priority, the first of these, as processed by the VT, shall become the active mask. When the active Working Set changes to a lower priority Alarm Mask or Data Mask, the next-highest priority Alarm Mask is processed in turn.

Operator input disturbed by the activation of an Alarm Mask may 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 mask can be active per Working Set.

b) The VT responds with an 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 acoustic signal shall be played. The VT shall terminate any acoustic signal from a lower priority Alarm Mask that can be in process. The VT shall terminate any acoustic signal from a control audio command that can be in process. Version 4 and later-version 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).

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

e) At some point, the operator acknowledges 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).

f) The Working Set may choose to ignore the key press if the ACK is not allowed or may 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 an Active Mask response.

h) The VT displays a new mask (see Table 3).

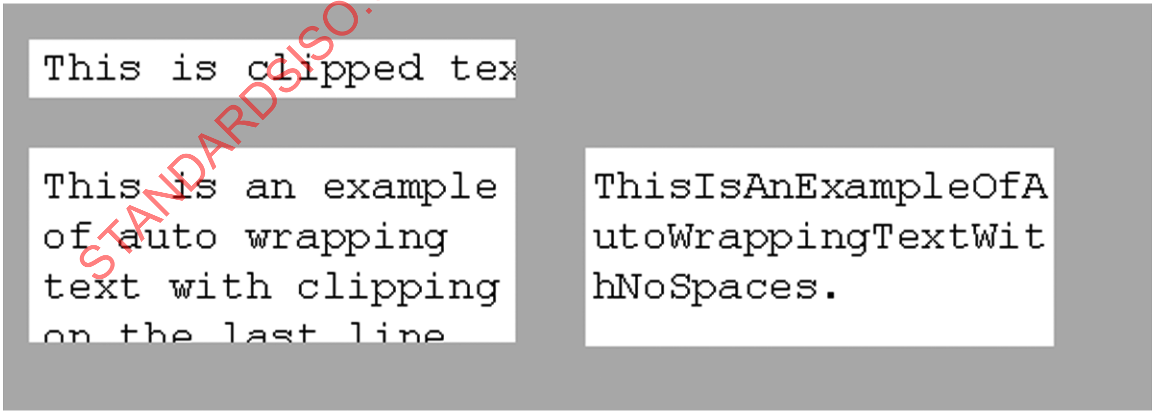
Table 3 — VT behaviour on mask transition

Working Set's active mask attribute from/to	Is requester current active Working Set?	VT behaviour
Data to data	Yes	Hide current Data Mask, show new Data Mask.
Data to data	No	No visual change.
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.
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.
Alarm to data	No	No visual change.

4.6.12 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 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 text is clipped on a graphical (i.e. pixel) basis (see Figure 11).



This is clipped tex

This is an example
of auto wrapping
text with clipping
on the last line

ThisIsAnExampleOfA
utoWrappingTextWit
hNoSpaces.

Figure 11 — Clipping examples

4.6.13 Scaling

4.6.13.1 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 may 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.

4.6.13.2 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). Dot size (number of pixels) should also be adjusted.

4.6.13.3 Fonts

Font scaling is not provided by the VT. 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. Working Set designers shall be aware of this limitation and shall take the necessary steps.

4.6.13.4 Picture graphic objects

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

4.6.14 Operator input

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

a) Navigating;

b) Data input.

NOTE 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 Data Mask;

— the object is wholly outside the visible area of a container, as defined by the container's width and height attributes.

Objects with width or height of zero, objects outside the Data Mask and objects outside a container's visible area are discouraged, as activation may not be possible (e.g. touch screen).

When a new active Data Mask is selected, the state is reset to “Navigating”. If an Alarm Mask is selected, then the VT may 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 ECU 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 4).

The initial focus point and the tab order for navigating to the various input fields or buttons is VT proprietary, but the ECU shall be aware that the tab order may be defined by the definition order of the input objects in the parent object.

The VT may choose not to send the VT Select Input Object message for every object that the operator passes through while navigating. For example, if the navigation means is a rotary control, the focus will change rapidly while the operator is spinning the control. In such a case, the VT may choose only to send the VT Select Input Object message to the input object that loses focus and that which eventually gets focus.

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

Examples of focus indicators include (but are 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 may 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 may 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 4).

Table 4 — VT reaction to navigation-related events

Current state	Command/Event	New state	Response frames[a]
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 may 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 the 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) Active Mask response VT Status message (with new 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)	Data input	Select Input Object response
Data input	Select Input Object command	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	(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	Data input	Change Numeric Value response (with error code indicating that the object is in use)
Data input	Change String Value command (transport protocol)	Data input	Change String Value response (with error code indicating that the object is in use)
Data input	Change Active Mask command (only if the 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 — if any) Active Mask response VT Status message (with new Data Mask) VT Select Input Object message (on object which gets focus — if any)

Table 4 (*continued*)

Current state	Command/Event	New state	Response frames[a]
Data input	Change Attribute command	Data input	Change Attribute response (with error code indicating that the object is in use)
Data input	Change List Item command	Data input	Change List Item response (with error code indicating that the object is in use)
Data input	Pool Update alters 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	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 Input String Value (even if the new value is the same as the old) VT Select Input Object message (on object which has/loses focus)

[a] They shall be sent in the indicated sequence.

4.6.15 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 Tables B.11 and B.13.

If a Key Object or non-latchable Button Object is erased from the screen (e.g. due to a Change Active Mask command, Soft Key Mask, Hide/Show Object command, etc.) while 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.

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.

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 key is pressed while a first key has already been detected as pressed/held, it shall be ignored.

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.

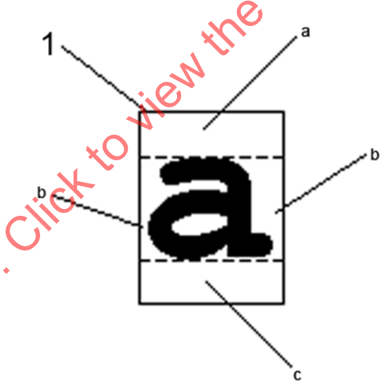
If a Key or Button is found to be pressed at power on, it shall not be reported as held; however, this can be cause for VT to report a diagnostic message to the operator.

4.6.16 Font rendering

4.6.16.1 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 12) 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

a Ascender area can create white space depending on VT design.

b Character rendering itself can create white space depending on VT design.

c Descender area can create white space depending on VT design.

Figure 12 — Graphical extents of a character

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

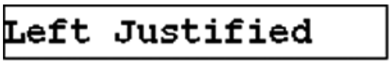
These modifications shall be done for visual justification — the stored value is not modified.

4.6.16.1.1 Horizontal left justification

When left-justified, the VT shall not remove any leading spaces; 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.

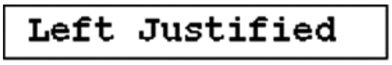
In the examples below, the box represents the extents of the defined text field area.

EXAMPLE 1 Left justification, no auto-wrap, no leading spaces.



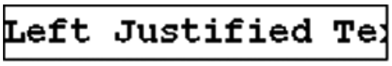
Left Justified

EXAMPLE 2 Left justification, no auto-wrap, one leading space.



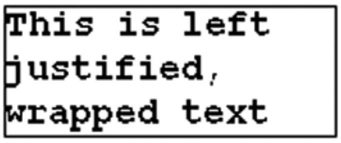
Left Justified

EXAMPLE 3 Left justification, no auto-wrap, clipping on the right.



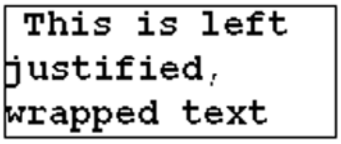
Left Justified Te

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



This is left
justified,
wrapped text

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



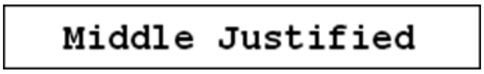
This is left
justified,
wrapped text

4.6.16.1.2 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.

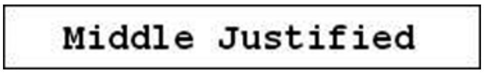
In the examples below, the box represents the extents of the defined text field area.

EXAMPLE 1 Middle justification, no auto-wrap, no leading or trailing spaces.



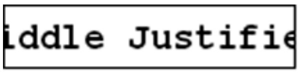
Middle Justified

EXAMPLE 2 Middle justification, no auto-wrap, one leading and one trailing space (space is removed).



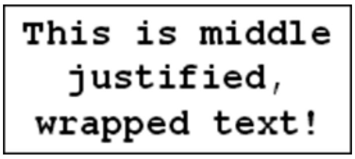
Middle Justified

EXAMPLE 3 Middle justification, no auto-wrap, clipping on the left and right.



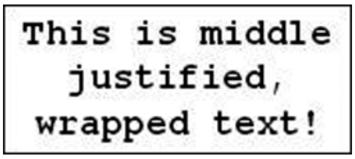
iddle Justifie

EXAMPLE 4 Middle justification, auto-wrap, no leading or trailing spaces.



This is middle
justified,
wrapped text!

EXAMPLE 5 Middle justification, auto-wrap, one leading and one trailing space (space is removed).



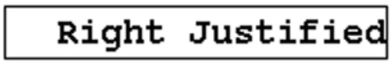
This is middle
justified,
wrapped text!

4.6.16.1.3 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.

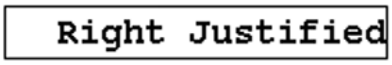
In the examples below, the box represents the extents of the defined text field area.

EXAMPLE 1 Right justification, no auto-wrap, no trailing spaces.



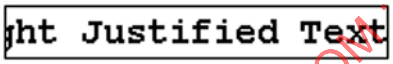
Right Justified

EXAMPLE 2 Right justification, no auto-wrap, one trailing space (space is removed).



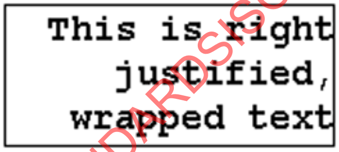
Right Justified

EXAMPLE 3 Right justification, no auto-wrap, clipping on the left.



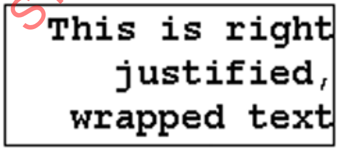
ght Justified Text

EXAMPLE 4 Right justification, auto-wrap, no trailing spaces.



This is right
justified,
wrapped text

EXAMPLE 5 Right justification, auto-wrap, one trailing space (space is removed).



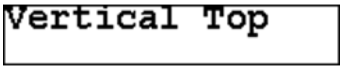
This is right
justified,
wrapped text

4.6.16.1.4 Vertical top justification

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.

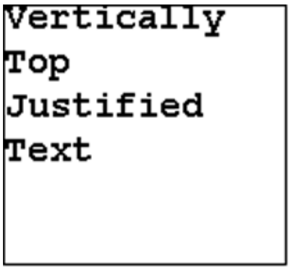
In the examples below, the box represents the extents of the defined text field area and the text is also left-justified.

EXAMPLE 1 Top justification, no auto-wrap.



Vertical Top

EXAMPLE 2 Top justification, auto-wrap.



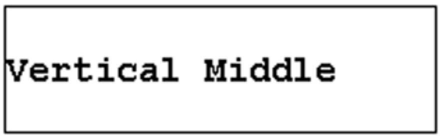
Vertically
Top
Justified
Text

4.6.16.1.5 Vertical middle justification

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.

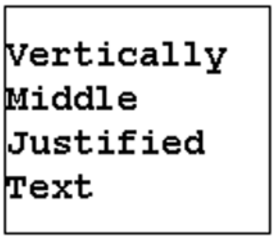
In the examples below, the box represents the extents of the defined text field area and the text is also left-justified.

EXAMPLE 1 Middle justification, no auto-wrap.



Vertical Middle

EXAMPLE 2 Middle justification, auto-wrap.



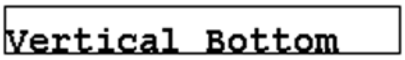
Vertically
Middle
Justified
Text

4.6.16.1.6 Vertical bottom justification

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.

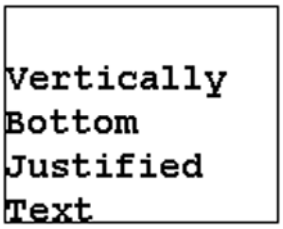
In the examples below, the box represents the extents of the defined text field area and the text is also left-justified.

EXAMPLE 1 Bottom justification, no auto-wrap.



Vertical Bottom

EXAMPLE 2 Bottom justification, auto-wrap.

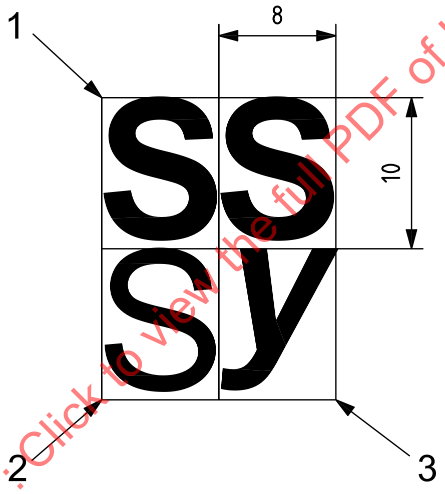


Vertically
Bottom
Justified
Text

4.6.16.2 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. An 8 × 10 font set to bold and italics, for example, 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 13).

Dimensions in pixels



Key

1 bold

2 normal (upright)

3 bold italic

Figure 13 — 8 × 10 fonts — Example

The VT designer may 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.

4.6.16.3 Proportional fonts

In Version 4 and later VTs as an option, the VT design may 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. 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.

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 report 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 one-pixel resolution. Due to characteristics of the implemented font rendering engine, and the shape of the characters, one-pixel resolution may not be detectable for every character, even though it is supported.

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 may 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 rule applies 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.

4.6.16.4 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 and height 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 there is no Space (20_{16}) in the line, then wrap on the last wholly visible character in the line (see Figure 11).
- At line end (see 4.6.16.5).

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

4.6.16.5 Non-printing characters in strings

Version 3 VTs and prior allowed CR, LF, BS and 0x00 control characters in strings. The rendered presentation was not precisely defined.

The following clarification applies to Version 4 VTs and later:

- BS (Back Space) is ignored, as are other characters denoted with scissors in Annex L;
- 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 may 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: ABCDEFCRGHIJKL or ABCDEFLFGHIJKL or ABCDEFCRLFGHIJKL	ABCDEF GHIJKL	ABCDEF GHIJKL	ABCDEF GHIJKL
String 4: MNOPQRLFCRSTUVWX	MNOPQR ¶ [a] STUVWX	MNOPQR ¶ [a] STUVWX	MNOPQR ¶ [a] STUVWX

[a] "¶" is shown for examples and would not be visible on the VT.

Figure 14 — CR and LF application to test strings

4.6.16.6 String encoding

Text strings can be encoded with either 8 bit characters (chars) or Unicode/ISO 10646 characters (WideChars).

VT Version 3 and prior supported font types are shown in Tables L.1 and L.2 .

VT Version 4 and later supported font types are shown in Tables L.1 to L.7.

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 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 ISO 11783-6 WideStrings 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 − 1

e.g. FF,FE,41,00,42,00,43,00 => “ABC”, [length = 8, number of WideChar = 3]

This is only correct if all characters are from Code Plane 0. The characters above $FFFF_{16}$ require 4 bytes per character (see “surrogate pair” below).

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

The VT may support any character defined by Unicode/ISO 10646, but as a minimum it shall support the characters listed in Table L.7.

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 may be VT proprietary (e.g. “□”).

The encoding of Input String 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:

— S = character − 10000_{16};

— High surrogate = $D800_{16}$ + (S shifted 10 bits to the right);

— 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 $DBFF_{16}$, $DFFF_{16}$.

4.6.17 Filling output shape objects

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).

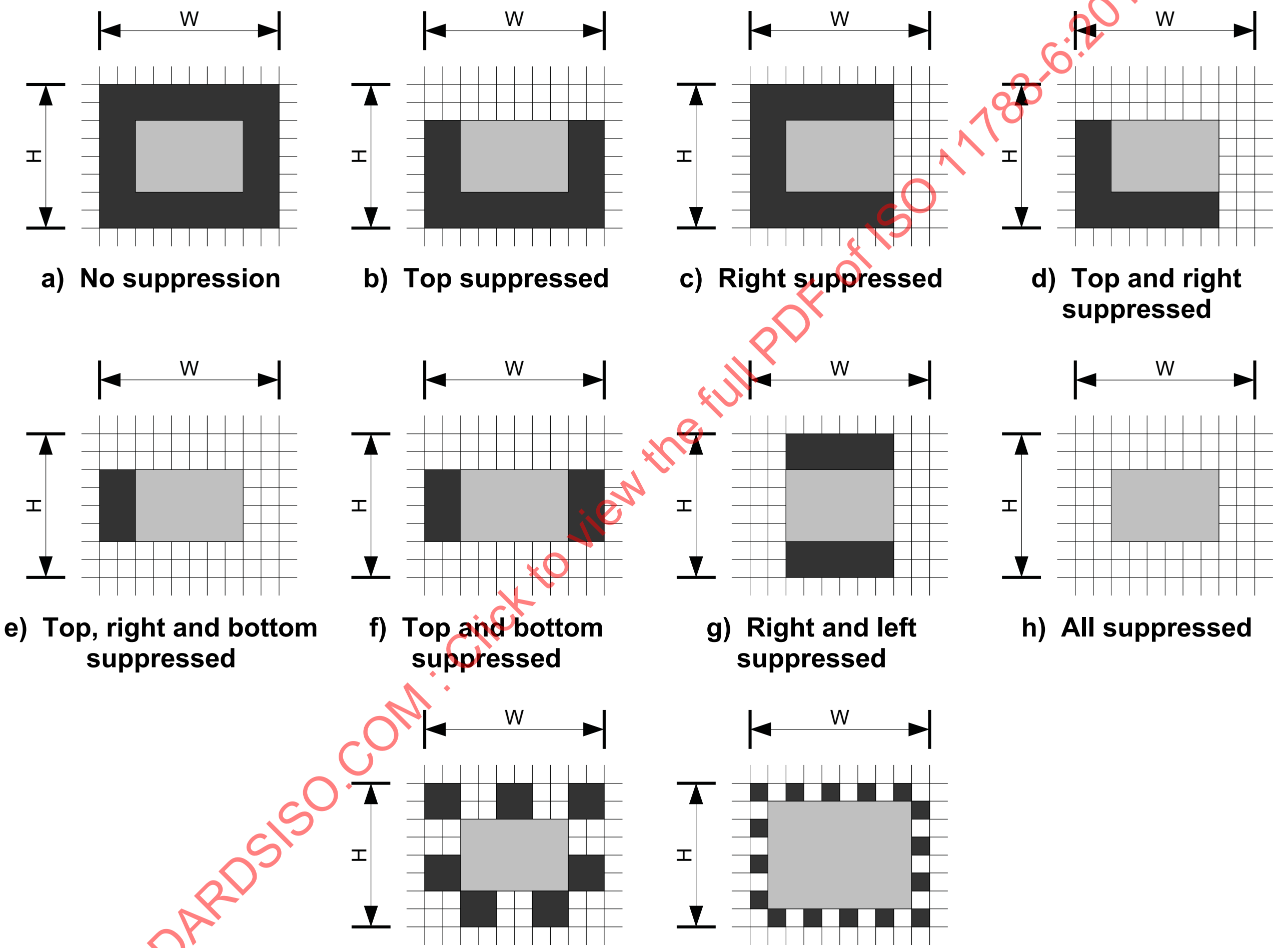
The following rules apply to Pattern fills made from Output Shape objects.

a) The pattern shall be a Picture Graphic object whose width is an integer divisible by 8. The raw data is used for the pattern and the object is not scaled regardless of the attributes of the object.

b) The upper left corner of the pattern buffer shall be anchored to the upper left corner of the VT's physical Data Mask Area and repeats across and also down. This ensures that the pattern matches between objects in the Data Mask Area and that the pattern fill looks the same on all VT designs.

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

Filling and line suppression examples are shown in Figures 15 to 17.



a) No suppression

b) Top suppressed

c) Right suppressed

d) Top and right suppressed

e) Top, right and bottom suppressed

f) Top and bottom suppressed

g) Right and left suppressed

h) All suppressed

i) No suppression

j) No suppression

Rectangles a) to h): width = 10, height = 8, line width = 2, fill type = solid grey.

Rectangle i): width = 10, height = 8, line width = 2, fill type = solid grey, line art 1010.

Rectangle j): width = 10, height = 8, line width = 1, fill type = solid grey, line art 1010.

Figure 15 — Rectangle line suppression and filling examples

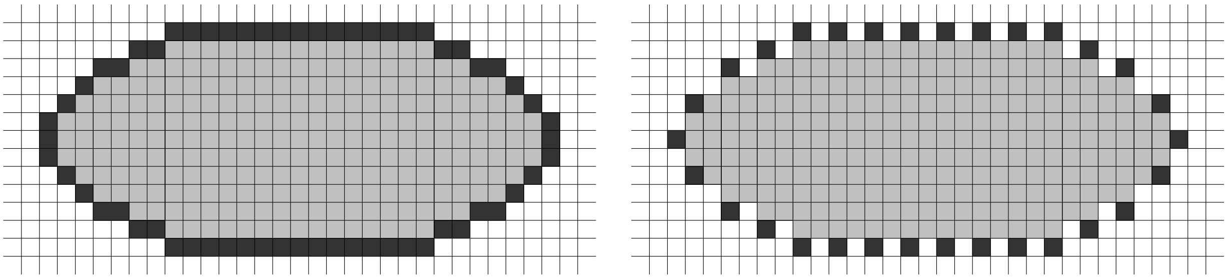


Figure 16 — Ellipse filling examples (without and with border line art)

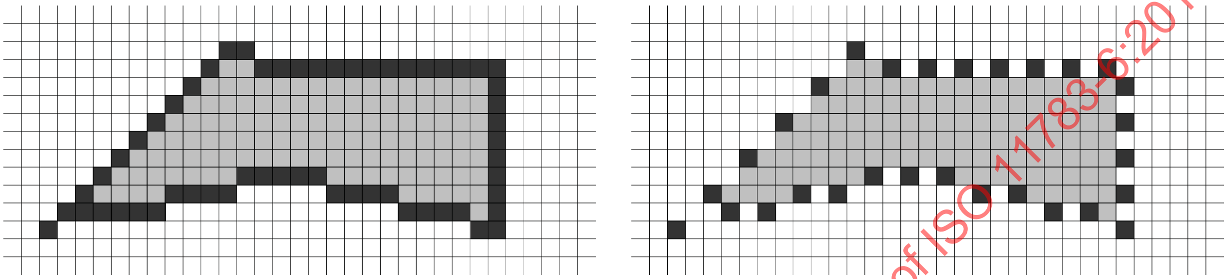


Figure 17 — Polygon filling examples (without and with border line art)

4.6.18 Events

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. 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 the other to control the audio signal.

EXAMPLE 3 A Change Numeric Value with an invalid value causes rejection of the command and an associated Macro is not executed. A Change Numeric Value 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.

4.6.19 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 *not associated* with a button or other input object has been touched or clicked on.

4.6.20 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 may 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.

4.6.21 VT Number

By default, VTs shall be factory set to function instance zero (0), but 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 VTs 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 VTs are not created. The new function instance shall not be used until a reboot of the VT is performed. The VT with function instance zero (0) is defined as the “primary VT”.

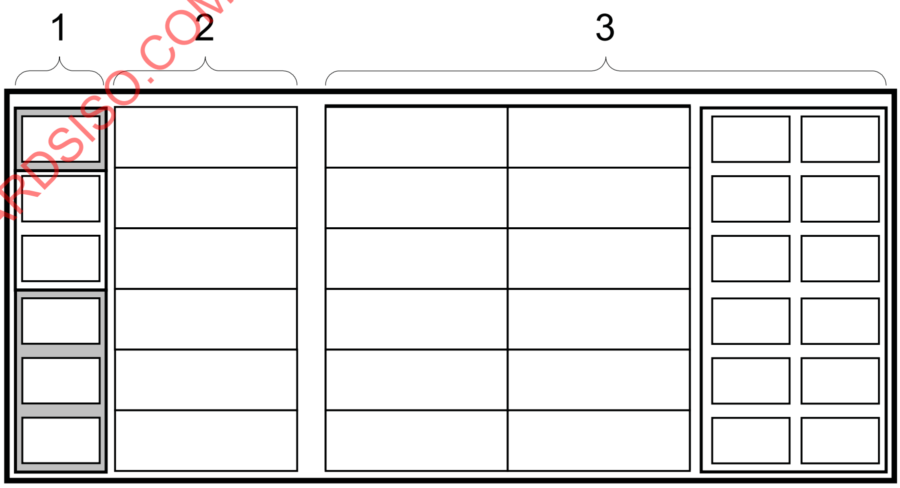
The proprietary means to set the function instance shall represent to the operator a VT Number. In this way, VTs from all manufacturers will present a consistent numbering scheme for the operator to choose the primary (and secondary) VT(s). To facilitate easy VT identification, the Identify VT message may be used.

4.7 Displaying data from Multiple Working Sets on one Mask (Version 4 VT and later)

4.7.1 General

The following subclauses are applicable to Version 4 and later VTs.

The VT may 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 VT's standard Data Mask and Soft Key Mask. See example in Figure 18.



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 18 — Displaying data from multiple Working Sets — Example

4.7.1.1 Minimum requirements

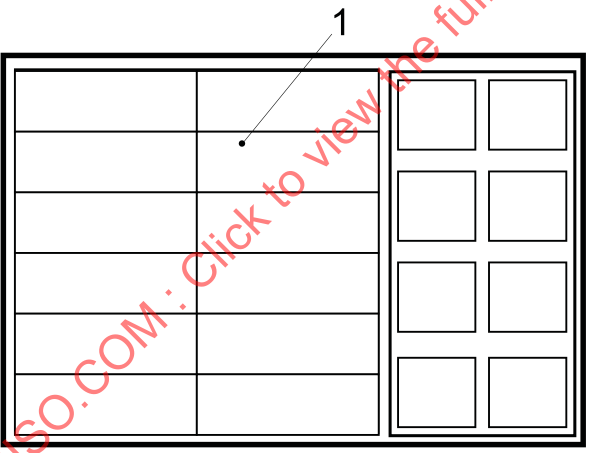
This is an optional feature. However, as a minimum, the VT shall be required to parse the Window Mask and Key Group objects even if not supported. This allows Working Sets to upload these objects to all Version 4 and later VTs without modification of the object pool. If this feature is supported, the 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 may also be supported as desired. Working Set designs could desire the use of any of the window mask types, so implementation by the VT design of all window mask types is recommended. 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 while ignoring 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 may 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 19).



Key

1 User-Layout Data Mask showing 12 window cells

Figure 19 — 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 may take up more than one Window Cell, and may 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 Figure 20).

NOTE Window Mask objects are not limited to the sizes shown in Figure 20, which is an example only.

4.7.4 Window Mask content

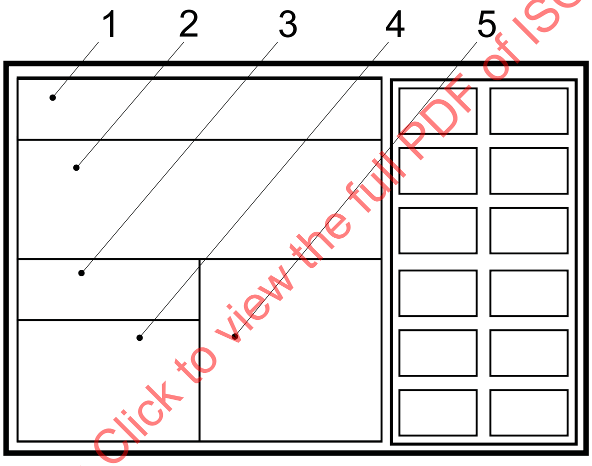
A Window Mask object can display content using two different types of presentations. This is controlled by the Working Set, using a Window Type attribute. See 4.7.15 for more details.

4.7.4.1 Working Set defined presentation

If the Window Mask object has a Window Type of zero, the Window Mask object behaves very similarly 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.

4.7.4.2 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

2 Window Mask, 2 × 2

3 Window Mask, 1 × 1

4 Window Mask, 1 × 2

5 Window Mask, 1 × 3

Figure 20 — 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 the 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 may occur at runtime.

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

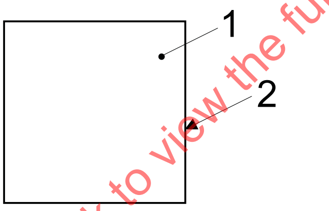
4.7.5 Window Cell Size and Borders

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 may 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 may 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 touch the border area when the free form (type 0) window type is used (see B.19.1). See Figure 21 for an example of the border.



Key

1 Window Mask region

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

Figure 21 — 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 200 × 200 Data Mask Area, using the equations given 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 Applying the Window Cell sizes above when using a VT with a default 200 × 200 Data Mask Area, a 2 × 2 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 Window Type is not the Free Form Window (0). The exception to this rule is the Window Icon and Button attributes that are pre-scaled by the Working Set. The VT may 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.19.1).

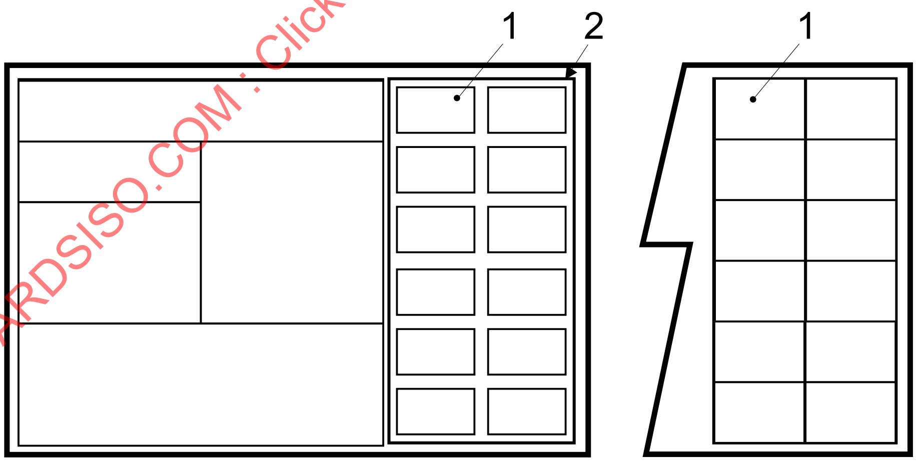
4.7.7 Using Window Masks outside User-Layout Data Masks

The VT may optionally use Window Mask objects in Non-VT Screens and Non-VT Areas, if supported by the manufacturer's design (see Figure 18). Soft Keys may or may not be supported when used in non-VT screens and non-VT areas. Working Sets shall be aware that Window Mask object may be used outside the User-Layout Data Mask and therefore shall monitor the VT On User-Layout Hide/Show message, refreshing 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 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 22.

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

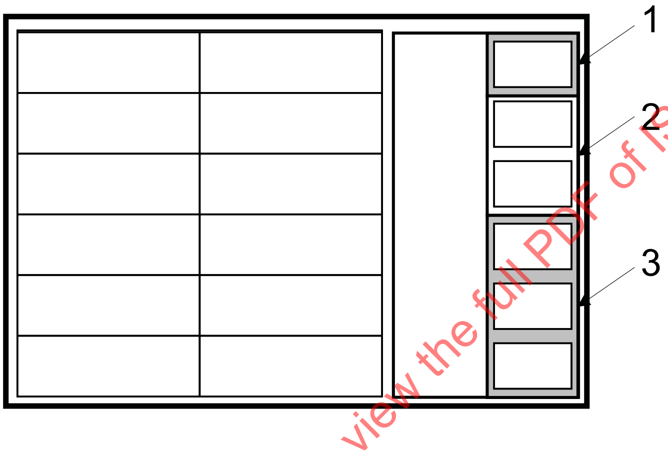
Figure 22 — Key Cell layout — Examples

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 may contain one to four Key Objects (although one is typical) and therefore may 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 may occur at runtime.

Working Set designers shall be aware that Keys may or may not be available; 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

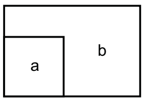
1 Key Group containing 1 Key

2 Key Group containing 2 Keys

3 Key Group containing 3 Keys

Figure 23 — User-Layout Data Mask with six Key Cells — Example

Working Set designers should ensure that Keys can be recognized to be part of a specific Working Set. For example, 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 24 is recommended.



[a] 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.

[b] Key Object in a Key Group: it is the size of a Soft Key designator and uses all remaining pixels in the standard designator area.

Figure 24 — Key Object in a Key Group indicating Working Set — Example

4.7.10 Key Cell Size and Borders

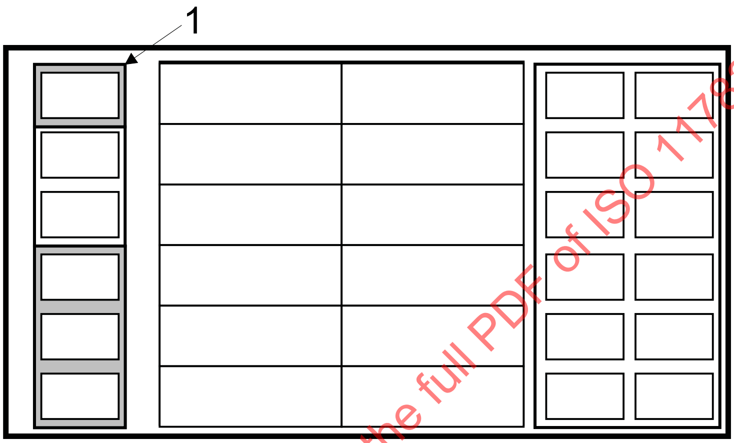
The same drawing rules that apply to Soft Key Mask objects also apply to User-Layout Soft Key Mask objects. A Key Cell size is the same size as a normal Soft Key designator.

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 User-Layout Soft Key Masks

The VT may optionally use Key Group objects outside the 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 the 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. See Figure 25.



Key

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

Figure 25 — Key Group Objects outside User-Layout Data Mask — Example

4.7.13 Operator Inputs

It is possible for input objects from several Working Sets to 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 which 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 must be aware that there may be no visible effect, since the Working Set is not the owner of the active mask.

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 which 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 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 could 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 can need to render the Window Mask objects but the VT decides what is displayed and where in the Window Mask object, assuming the window type is greater than zero. The following guidelines apply.

a) Working Set designers should make the Window Mask objects and Key Group objects transparent. Therefore, the background colour is chosen by the VT. If required, the Working Set designer may query the VT's background colour with the Get Window Mask Data message.

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

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

4.7.15.1 Window Title and Window Title Font Attributes

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

4.7.15.2 Window Icon size and shape

The VT may optionally use the Window Icon attribute inside the Window Mask. The Window Icon attribute in the Window Mask object shall reference a Picture Graphic object. Positioning is proprietary to the VT designer, subject to the rules given 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 200 × 200 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 × 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 may pre-scale the Window Mask Icon Picture Graphic object at design time or set the scale, via the width attribute, at runtime. 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 may 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 of 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 is evident to the operator.

4.7.15.3 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 may be ignored by the VT in achieving the desired look and feel. More information on field lengths is given in the section on Window Mask Window Type (see B.19.1).

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 runtime is permitted by Annex C. 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 5.

Table 5 — 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 it is 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 not, the VT shall automatically remove the window from the stored mapping and, if visible, shall blank 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.
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 it is removed from the screen. The positions of other mapped Key Group objects shall not be affected.
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) the 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 affected. If the number of keys in the group increases, the VT shall determine whether 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 not, the VT shall automatically remove the Key Group from the mapping and, if the object is visible, shall blank the key positions that it originally occupied. The positions of other mapped Key Group objects shall not be affected.

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. 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 Field	7_{10}	Used to input a TRUE/FALSE type input.
Input String Field	8_{10}	Used to input a character string.
Input Number Field	9_{10}	Used to input an integer or float numeric.
Input List Field	10_{10}	Used to select an item from a pre-defined list.
Output field objects		
Output String Field	11_{10}	Used to output a character string.
Output Number Field	12_{10}	Used to output an integer or float numeric.
Output List object[a]	37_{10}	Used to output a list item.

Table A.1 (*continued*)

Object	Type ID	Description
Output shape objects		
Line	13_{10}	Used to output a line.
Rectangle object	14_{10}	Used to output a rectangle or square.
Ellipse	15_{10}	Used to output an ellipse or circle.
Polygon	16_{10}	Used to output a polygon.
Output graphic objects		
Meter object	17_{10}	Used to output a meter.
Linear Bar Graph object	18_{10}	Used to output a linear bar graph.
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.
Picture graphic object		
Picture Graphic object	20_{10}	Used to output a picture graphic (bitmap).
Variable objects		
Number Variable	21_{10}	Used to store a 32 bit unsigned integer value.
String Variable	22_{10}	Used to store a fixed length string value.
Attribute objects		
Font Attributes	23_{10}	Used to group font-based attributes. Can only be referenced by other objects.
Line Attributes	24_{10}	Used to group line-based attributes. Can only be referenced by other objects.
Fill Attributes	25_{10}	Used to group fill-based attributes. Can only be referenced by other objects.
Input Attributes	26_{10}	Used to specify a list of valid characters. Can only be referenced by input field objects.
Extended Input Attributes[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	40_{10}	Provides a mechanism to assign a String Variable and/or graphical designator as a label to other objects.
Pointer object		
Object Pointer	27_{10}	Used to reference another object.
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 VTs may use the Execute Macro command.

Table A.1 (*continued*)

Object	Type ID	Description
Auxiliary control		
Auxiliary Function Type 1 object (ignored)[c]	29_{10}	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}	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}	Defines the designator and function type for an Auxiliary Function.
Auxiliary Input Type 2 object[b]	32_{10}	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.
Not used	41_{10} to 239_{10}	Reserved.
Proprietary objects		
Manufacturer defined objects[d]	240_{10} to 254_{10}	Manufacturer defined objects should not be sent to any other vendor's VT (see 4.6.20).
Reserved objects		
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).

A.1.2 Nomenclature

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

[ ] when surrounding an AID number, indicates a read-only attribute;

Array sequence of 1 byte unsigned integer values of a defined length;

Bitmask set of logical bit values, size is 1 byte, with Bit 0 always defined as the least significant bit (see Figure A.1);

Boolean logical TRUE (1) or FALSE (0), size is 1 byte;

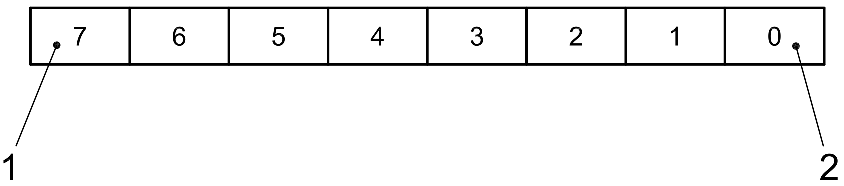
Byte signed or unsigned integer numeric value with a size of exactly 1 byte;

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 the primitive type “char”. 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 visible objects of an object pool are arranged in a hierarchy, where parent objects contain child objects. Some of the child objects can also contain objects, and can then 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

		Parent object																
		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
Child Objects	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
	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

Table A.2 (*continued*)

	Parent object																
	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
Output Number object	2	2	2	2	4		2	2		2	4	2	2	3	3	4	4
Output List object	4	4	4	4	4		4	4		4	4			4	4	4	4
Line object	2	2	2	2	4		2	2		4	4	2	2	3	3	4	4
Rectangle object	2	2	2	2	4		2	2		4	4	2	2	3	3	4	4
Ellipse object	2	2	2	2	4		2	2		4	4	2	2	3	3	4	4
Polygon object	2	2	2	2	4		2	2		4	4	2	2	3	3	4	4
Meter object	4	2	2	2	4		4	4		4	4			3	3	4	4
Linear Bar Graph object	4	2	2	2	4		4	4		4	4			3	3	4	4
Arched Bar Graph object	4	2	2	2	4		4	4		4	4			3	3	4	4
Picture Graphic object	2	2	2	2	4		2	2		2	4	2	2	3	3	4	4
Graphics Context object	4	4	4	4	4		4	4		4	4			4	4	4	4
Number Variable object																	
String Variable object																	
Font Attributes object																	
Line Attributes object																	
Fill Attributes object																	
Input Attributes object																	
Extended Input Attributes object																	
Object Pointer object	4	2	2	2	4	2	2	2	4	4	4			3	3	4	
Macro object																	
Colour Map object																	
Auxiliary Function Type 1 object																	
Auxiliary Input Type 1 object																	
Auxiliary Function Type 2 object		3	3	3													4
Auxiliary Input Type 2 object		3	3	3													4
Auxiliary Control Designator Type 2 Object Pointer		3	3	3	3						4						

NOTE 1 The numbers 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	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 drawn.
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).
On disable	6_{10}	Input object is disabled (only enabled input objects can be navigated to).
On Change Active Mask	7_{10}	Change Active Mask command.
On Change Soft Key Mask	8_{10}	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}	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 or Change String Value command (transport protocol).
On Change Priority	17_{10}	Priority command.
On Change End Point	18_{10}	Change End Point command.
On Input Field Selection	19_{10}	The input field or Button has received focus, operator has navigated onto the input field or Select Input Object command.
On Input Field Deselection	20_{10}	The input field or Button has lost focus, operator has navigated off the input field or 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}	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 (transport protocol).
Reserved	27_{10}–239_{10}	Reserved.
Proprietary Events	240_{10}–254_{10}	Proprietary events should not be used between ECUs and VTs with different manufacturer codes.
Reserved	255_{10}	Reserved.

[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 OnEntryOfANewValue and/or OnEntryOfAValue Macros are executed for the modified input object *only*.

A.3 VT standard colour palette

The VT Standard colour palette is shown in Table A.4. The active Colour Map can be altered (see F.60).

Table A.4 — Standard VT RGB colour palette

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

Table A.4 (*continued*)

Index	R,G,B value	Index	R,G,B value	Index	R,G,B value	Index	R,G,B value
34	00,99,00	98	66,33,CC	162	CC,00,66	226	FF,FF,00
35	00,99,33	99	66,33,FF	163	CC,00,99	227	FF,FF,33
36	00,99,66	100	66,66,00	164	CC,00,CC	228	FF,FF,66
37	00,99,99	101	66,66,33	165	CC,00,FF	229	FF,FF,99
38	00,99,CC	102	66,66,66	166	CC,33,00	230	FF,FF,CC
39	00,99,FF	103	66,66,99	167	CC,33,33	231	FF,FF,FF
40	00,CC,00	104	66,66,CC	168	CC,33,66	232	Proprietary
41	00,CC,33	105	66,66,FF	169	CC,33,99	233	Proprietary
42	00,CC,66	106	66,99,00	170	CC,33,CC	234	Proprietary
43	00,CC,99	107	66,99,33	171	CC,33,FF	235	Proprietary
44	00,CC,CC	108	66,99,66	172	CC,66,00	236	Proprietary
45	00,CC,FF	109	66,99,99	173	CC,66,33	237	Proprietary
46	00,FF,00	110	66,99,CC	174	CC,66,66	238	Proprietary
47	00,FF,33	111	66,99,FF	175	CC,66,99	239	Proprietary
48	00,FF,66	112	66,CC,00	176	CC,66,CC	240	Proprietary
49	00,FF,99	113	66,CC,33	177	CC,66,FF	241	Proprietary
50	00,FF,CC	114	66,CC,66	178	CC,99,00	242	Proprietary
51	00,FF,FF	115	66,CC,99	179	CC,99,33	243	Proprietary
52	33,00,00	116	66,CC,CC	180	CC,99,66	244	Proprietary
53	33,00,33	117	66,CC,FF	181	CC,99,99	245	Proprietary
54	33,00,66	118	66,FF,00	182	CC,99,CC	246	Proprietary
55	33,00,99	119	66,FF,33	183	CC,99,FF	247	Proprietary
56	33,00,CC	120	66,FF,66	184	CC,CC,00	248	Proprietary
57	33,00,FF	121	66,FF,99	185	CC,CC,33	249	Proprietary
58	33,33,00	122	66,FF,CC	186	CC,CC,66	250	Proprietary
59	33,33,33	123	66,FF,FF	187	CC,CC,99	251	Proprietary
60	33,33,66	124	99,00,00	188	CC,CC,CC	252	Proprietary
61	33,33,99	125	99,00,33	189	CC,CC,FF	253	Proprietary
62	33,33,CC	126	99,00,66	190	CC,FF,00	254	Proprietary
63	33,33,FF	127	99,00,99	191	CC,FF,33	255	Proprietary

[a] Monochrome (0 and 1).

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

The VT 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 for extending the colour palette. 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 Table A.4.

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	Direction	Function (Decimal)	Function (Hex)	Allowed in Macro
C.2.3	Object pool transfer message (transport protocol)	ECU to VT	17_{10}	11_{16}	No
C.2.4	End of Object Pool message	ECU to VT	18_{10}	12_{16}	No
C.2.5	End of Object Pool response	VT to ECU	18_{10}	12_{16}	No
D.2	Get Memory	ECU to VT	192_{10}	$C0_{16}$	No
D.3	Get Memory response	VT to ECU	192_{10}	$C0_{16}$	No
D.4	Get Number of Soft Keys message	ECU to VT	194_{10}	$C2_{16}$	No
D.5	Get Number of Soft Keys response	VT to ECU	194_{10}	$C2_{16}$	No
D.6	Get Text Font Data message	ECU to VT	195_{10}	$C3_{16}$	No
D.7	Get Text Font Data response	VT to ECU	195_{10}	$C3_{16}$	No
D.8	Get Hardware message	ECU to VT	199_{10}	$C7_{16}$	No
D.9	Get Hardware response	VT to ECU	199_{10}	$C7_{16}$	No
D.10	Get Supported Widechars message	ECU to VT	193_{10}	$C1_{16}$	No
D.11	Get Supported WideChars response (transport protocol)	VT to ECU	193_{10}	$C1_{16}$	No
D.12	Get Window Mask Data message	ECU to VT	196_{10}	$C4_{16}$	No
D.13	Get Window Mask Data response	VT to ECU	196_{10}	$C4_{16}$	No

Table A.5 (*continued*)

Clause/ subclause	Message	Direction	Function (Decimal)	Function (Hex)	Allowed in Macro
D.14	Get Supported Objects message	ECU to VT	197_{10}	$C5_{16}$	No
D.15	Get Supported Objects response	VT to ECU	197_{10}	$C5_{16}$	No
E.2	Get Versions message	ECU to VT	223_{10}	DF_{16}	No
E.3	Get Versions response (transport protocol)	VT to ECU	224_{10}	$E0_{16}$	No
E.4	Store Version command	ECU to VT	208_{10}	$D0_{16}$	No
E.5	Store Version response	VT to ECU	208_{10}	$D0_{16}$	No
E.6	Load Version command	ECU to VT	209_{10}	$D1_{16}$	No
E.7	Load Version response	VT to ECU	209_{10}	$D1_{16}$	No
E.8	Delete Version command	ECU to VT	210_{10}	$D2_{16}$	No
E.9	Delete Version response	VT to ECU	210_{10}	$D2_{16}$	No
F.2	Hide/Show Object command	ECU to VT	160_{10}	$A0_{16}$	Yes
F.3	Hide/Show Object response	VT to ECU	160_{10}	$A0_{16}$	No
F.4	Enable/Disable Object command	ECU to VT	161_{10}	$A1_{16}$	Yes
F.5	Enable/Disable Object response	VT to ECU	161_{10}	$A1_{16}$	No
F.6	Select Input Object command	ECU to VT	162_{10}	$A2_{16}$	Yes
F.7	Select Input Object response	VT to ECU	162_{10}	$A2_{16}$	No
F.8	ESC command	ECU to VT	146_{10}	92_{16}	No
F.9	ESC response	VT to ECU	146_{10}	92_{16}	No
F.10	Control Audio Signal command	ECU to VT	163_{10}	$A3_{16}$	Yes
F.11	Control Audio Signal response	VT to ECU	163_{10}	$A3_{16}$	No
F.12	Set Audio Volume command	ECU to VT	164_{10}	$A4_{16}$	Yes
F.13	Set Audio Volume response	VT to ECU	164_{10}	$A4_{16}$	No
F.14	Change Child Location command	ECU to VT	165_{10}	$A5_{16}$	Yes
F.15	Child Location response	VT to ECU	165_{10}	$A5_{16}$	No
F.16	Change Child Position command (transport protocol)	ECU to VT	180_{10}	$B4_{16}$	Yes
F.17	Change Child Position response	VT to ECU	180_{10}	$B4_{16}$	No
F.18	Change Size command	ECU to VT	166_{10}	$A6_{16}$	Yes
F.19	Change Size response	VT to ECU	166_{10}	$A6_{16}$	No
F.20	Change Background Colour command	ECU to VT	167_{10}	$A7_{16}$	Yes
F.21	Change Background Colour response	VT to ECU	167_{10}	$A7_{16}$	No
F.22	Change Numeric Value command	ECU to VT	168_{10}	$A8_{16}$	Yes
F.23	Change Numeric Value response	VT to ECU	168_{10}	$A8_{16}$	No
F.24	Change String Value command (transport protocol)	ECU to VT	179_{10}	$B3_{16}$	Yes
F.25	Change String Value response	VT to ECU	179_{10}	$B3_{16}$	No

Table A.5 (*continued*)

Clause/ subclause	Message	Direction	Function (Decimal)	Function (Hex)	Allowed in Macro
F.26	Change End Point command	ECU to VT	169_{10}	$A9_{16}$	Yes
F.27	Change End Point response	VT to ECU	169_{10}	$A9_{16}$	No
F.28	Change Font Attributes command	ECU to VT	170_{10}	AA_{16}	Yes
F.29	Change Font Attributes response	VT to ECU	170_{10}	AA_{16}	No
F.30	Change Line Attributes command	ECU to VT	171_{10}	AB_{16}	Yes
F.31	Change Line Attributes response	VT to ECU	171_{10}	AB_{16}	No
F.32	Change Fill Attributes command	ECU to VT	172_{10}	AC_{16}	Yes
F.33	Change Fill Attributes response	VT to ECU	172_{10}	AC_{16}	No
F.34	Change Active Mask command	ECU to VT	173_{10}	AD_{16}	Yes
F.35	Change Active Mask response	VT to ECU	173_{10}	AD_{16}	No
F.36	Change Soft Key Mask command	ECU to VT	174_{10}	AE_{16}	Yes
F.37	Change Soft Key Mask response	VT to ECU	174_{10}	AE_{16}	No
F.38	Change Attribute command	ECU to VT	175_{10}	AF_{16}	Yes
F.39	Change Attribute response	VT to ECU	175_{10}	AF_{16}	No
F.40	Change Priority command	ECU to VT	176_{10}	$B0_{16}$	Yes
F.41	Change Priority response	VT to ECU	176_{10}	$B0_{16}$	No
F.42	Change List Item command	ECU to VT	177_{10}	$B1_{16}$	Yes
F.43	Change List Item response	VT to ECU	177_{10}	$B1_{16}$	No
F.44	Delete Object Pool command	ECU to VT	178_{10}	$B2_{16}$	No
F.45	Delete Object Pool response	VT to ECU	178_{10}	$B2_{16}$	No
F.46	Lock/Unlock Mask command	ECU to VT	189_{10}	BD_{16}	Yes
F.47	Lock/Unlock Mask response	VT to ECU	189_{10}	BD_{16}	No
F.48	Execute Macro command	ECU to VT	190_{10}	BE_{16}	Yes
F.49	Execute Macro response	VT to ECU	190_{10}	BE_{16}	No
F.50	Change Object Label command	ECU to VT	181_{10}	$B5_{16}$	Yes
F.51	Change Object Label response	VT to ECU	181_{10}	$B5_{16}$	No
F.52	Change Polygon Point command	ECU to VT	182_{10}	$B6_{16}$	Yes
F.53	Change Polygon Point response	VT to ECU	182_{10}	$B6_{16}$	No
F.54	Change Polygon Scale command	ECU to VT	183_{10}	$B7_{16}$	Yes
F.55	Change Polygon Scale response	VT to ECU	183_{10}	$B7_{16}$	No
F.56	Graphics Context command (transport protocol)	ECU to VT	184_{10}	$B8_{16}$	Yes
F.57	Graphics Context response	VT to ECU	184_{10}	$B8_{16}$	No
F.58	Get Attribute Value message	ECU to VT	185_{10}	$B9_{16}$	No

Table A.5 (*continued*)

Clause/ subclause	Message	Direction	Function (Decimal)	Function (Hex)	Allowed in Macro
F.59	Get Attribute Value response	VT to ECU	185_{10}	$B9_{16}$	No
F.60	Select Colour Map command	ECU to VT	186_{10}	BA_{16}	No
F.61	Select Colour Map response	VT to ECU	186_{10}	BA_{16}	No
F.62	Identify VT message	ECU to GA, ECU to VT	187_{10}	BB_{16}	No
F.63	Identify VT response	VT to ECU	187_{10}	BB_{16}	No
G.2	VT Status message	VT to GA	254_{10}	FE_{16}	No
G.3	Working Set Maintenance message	ECU to VT	255_{10}	FF_{16}	No
H.2	Soft Key Activation message	VT to ECU	0_{10}	00_{16}	No
H.3	Soft Key Activation response	ECU to VT	0_{10}	00_{16}	No
H.4	Button Activation message	VT to ECU	1_{10}	01_{16}	No
H.5	Button Activation response	ECU to VT	1_{10}	01_{16}	No
H.6	Pointing Event message	VT to ECU	2_{10}	02_{16}	No
H.7	Pointing Event response	ECU to VT	2_{10}	02_{16}	No
H.8	VT Select Input Object message	VT to ECU	3_{10}	03_{16}	No
H.9	VT Select Input Object response	ECU to VT	3_{10}	03_{16}	No
H.10	VT ESC message	VT to ECU	4_{10}	04_{16}	No
H.11	VT ESC response	ECU to VT	4_{10}	04_{16}	No
H.12	VT Change Numeric Value message	VT to ECU	5_{10}	05_{16}	No
H.13	VT Change Numeric Value response	ECU to VT	5_{10}	05_{16}	No
H.14	VT Change Active Mask message	VT to ECU	6_{10}	06_{16}	No
H.15	VT Change Active Mask response	ECU to VT	6_{10}	06_{16}	No
H.16	VT Change Soft Key Mask message	VT to ECU	7_{10}	07_{16}	No
H.17	VT Change Soft Key Mask response	ECU to VT	7_{10}	07_{16}	No
H.18	VT Change String Value message (transport protocol)	VT to ECU	8_{10}	08_{16}	No
H.19	VT Change String Value response	ECU to VT	8_{10}	08_{16}	No
H.20	VT On User-Layout Hide/Show message	VT to ECU	9_{10}	09_{16}	No
H.21	VT On User-Layout Hide/Show response	ECU to VT	9_{10}	09_{16}	No
H.22	VT Control Audio Signal Termination message	VT to ECU	10_{10}	$0A_{16}$	No
J.7.2	Auxiliary Assignment Type 1 command	VT to ECU	32_{10}	20_{16}	No
J.7.3	Auxiliary Assignment Type 1 response	ECU to VT	32_{10}	20_{16}	No

Table A.5 (*continued*)

Clause/ subclause	Message	Direction	Function (Decimal)	Function (Hex)	Allowed in Macro
J.7.4	Auxiliary Input Type 1 status	ECU to GA	33_{10}	21_{16}	No
J.7.5	Auxiliary Assignment Type 2 command	VT to ECU	36_{10}	24_{16}	No
J.7.6	Auxiliary Assignment Type 2 response	ECU to VT	36_{10}	24_{16}	No
J.7.7	Preferred Assignment command	ECU to VT	34_{10}	22_{16}	No
J.7.8	Preferred Assignment response	VT to ECU	34_{10}	22_{16}	No
J.7.9	Auxiliary Input Type 2 Status message	ECU to GA	38_{10}	26_{16}	No
J.7.10	Auxiliary Input Type 2 Maintenance message	ECU to GA	35_{10}	23_{16}	No
J.7.11	Auxiliary Input Status Type 2 Enable command	VT to ECU	37_{10}	25_{16}	No
J.7.12	Auxiliary Input Status Type 2 Enable response	ECU to VT	37_{10}	25_{16}	No
Special functions					
	Reserved[a]	ECU to VT	10_{10}	$0A_{16}$	
	Reserved[a]	Any	11_{10} to 16_{10}	$0B_{16}$ to 10_{16}	
	Reserved[a]	Any	19_{10} to 31_{10}	13_{16} to $1F_{16}$	
	Reserved[a]	Any	36_{10}	24_{16}	
	Reserved[a]	Any	38_{10} to 95_{10}	26_{16} to $5F_{16}$	
	Reserved[a]	Any	128_{10} to 145_{10}	80_{16} to 91_{16}	
	Reserved[a]	Any	147_{10} to 159_{10}	93_{16} to $9F_{16}$	
	Reserved[a]	Any	188_{10}	BC_{16}	
	Reserved[a]	Any	191_{10}	BF_{16}	
	Reserved[a]	Any	198_{10}	$C6_{16}$	
	Reserved[a]	Any	200_{10} to 207_{10}	$C8_{16}$ to CF_{16}	
	Reserved[a]	Any	211_{10} to 222_{10}	$D3_{16}$ to DE_{16}	
	Reserved[a]	Any	225_{10} to 253_{10}	$E1_{16}$ to FD_{16}	
	Proprietary command[b]	Any	96_{10} to 127_{10}	60_{16} to $7F_{16}$	

[a] Reserved for future use.

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

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 may 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 Tables B.1 and B.2.

Allowed commands:

- Change Active Mask command;
- Change Background Colour command;
- Change Child Location command;
- Change Child Position command (transport protocol).

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.	—
On Change Active Mask	Change Active Mask command	If this Working Set is active, then perform a hide event on the current Data Mask and a show event on the new mask. Otherwise, just change the active mask attribute.	Active Mask response
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	Draw child object at current location in background colour to erase it. Refresh Data Mask (to redraw child object or objects).	Child Location response
On Change Child Position	Change Child Position command (transport protocol)	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

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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=0	3	Object Type = Working Set
Background colour	[1]	Integer	1	0 to 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 (e.g. auxiliary input units that do not have any Data Masks should not be made selectable).
Active mask	[3]	Integer	2	0 to 65534	6–7	The object ID of the data or Alarm Mask to display whenever the Working Set is active. If the selectable attribute value = 0, then the mask data is ignored.
Number of objects to follow		Integer	1	1 to 255	8	The objects that follow are used as the Working Set identifier. Although the identifier may 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 to 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.
Number of languages to follow		Integer	1	0 to 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 to 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 a Soft Key designator).
{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 a Soft Key designator).
Repeat: {Event ID}		Integer	1	0 to 255	11+(No. objects *6)…	(List these after all objects have been listed.) Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	12+(No. objects *6)…	Macro ID of the Macro to execute.
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 to z and A to 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. Since only one Data Mask can be displayed at a time, it is assumed that the entire Data Mask area of the physical display is filled. For this reason, no size attribute is required. See Tables B.3 and B.4.

Allowed commands:

- Change Background Colour command;
- Change Child Location command;
- Change Child Position command (transport protocol);
- Change Soft Key Mask command;
- Change Attribute command.

Table B.3 — Data Mask events

Event	Caused by	VT behaviour	Message
On Show	Both the Data Mask and its Working Set becoming active	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	Either the active mask changed for the Working Set or the Working Set being deactivated	Hide associated Soft Key Mask of this Data Mask.	—
On Refresh	Any action that causes a show or hide 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 (transport protocol)	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.

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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=1	3	Object Type = Data Mask
Background colour	1	Integer	1	0 to 255	4	Background colour
Soft Key Mask	2	Integer	2	0 to 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 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 to 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.
Number of macros to follow		Integer	1	0 to 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.
Repeat: {Object ID}		Integer	2	0 to 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 to 255	9+(No. objects *6)…	(List these after all objects have been listed.) Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	10+(No. objects *6)…	Macro ID of the Macro to execute.

B.3 Alarm Mask object

For information on Alarm Mask behaviour, see 4.6.11. See Tables B.5 and B.6.

Allowed commands:

— Change Background Colour command;

— Change Child Location command;

— Change Child Position command (transport protocol);

- Change Priority command;
- Change Soft Key Mask command;
- Change Attribute command.

Table B.5 — Alarm Mask events

Event	Caused by	VT behaviour	Message
On Show	Both the Alarm Mask and its Working Set become active.	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	Either the active mask is changed for the Working Set or the Working Set is deactivated.	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 (transport protocol)	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 re-evaluate 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 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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=2	3	Object Type = Alarm Mask
Background colour	1	Integer	1	0 to 255	4	Background colour.
Soft Key Mask	2	Integer	2	0 to 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 NULL Object ID 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 to 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 to 3	8	Acoustic signal. 0 = highest priority, 1 = medium priority, 2 = lowest priority, 3 = none (silent).
Number of objects to follow		Integer	1	0 to 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 to 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.
Repeat: {Object ID}		Integer	2	0 to 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).
{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 to 255	11+(No. objects *6)…	(List these after all objects have been listed.) Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	12+(No. objects *6)…	Macro ID of the Macro to execute.

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 runtime under Working Set control. The Container object has defined size limits to assist in determining when other objects are overlaid with the container. See Tables B.7 and B.8.

Allowed commands:

- Hide/Show Object command
- Change Child Location command
- Change Child Position command (transport protocol)
- Change Size command

Table B.7 — Container events

Event	Caused by	VT behaviour	Message
On Show	Show command.	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.	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 (transport protocol)	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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=3	3	Object Type = Container
Width	[1]	Integer	2	0 to 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 to 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 to 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 to 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.
Repeat: {Object ID}		Integer	2	0 to 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 to 255	11+(No. objects *6)…	(List these after all objects have been listed.) Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	12+(No. objects *6)…	Macro ID of the Macro to execute.

B.5 Soft Key Mask object

The Soft Key Mask is a Container object that contains Key Objects and pointers to Key Objects only. 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 Tables B.9 and B.10.

A common implementation, which was not clearly defined until VT Version 4 and later, is that Pointers to 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 Object ID 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 may be dynamic at runtime based if pointer values are changed to and from NULL Object ID (see 4.5.2.4).

Allowed commands:

- Change Background Colour command;
- Change Attribute command.

Table B.9 — Soft Key Mask events

Event	Caused by	VT behaviour	Message
On Show	Show parent Alarm/Data Mask or Soft Key Mask	Draw child objects in the order they are listed in the Soft Key Mask.	—
On Hide	Hide on parent Alarm/Data Mask or Soft Key Mask	—	—
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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=4	3	Object Type = Soft Key Mask
Background colour	1	Integer	1	0 to 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 to 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 to 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.
Repeat: {Object ID}		Integer	2	0 to 65534	7+object*2…	Object ID of a key or Object Pointer contained in this mask (see A.1.3).
Repeat: {Event ID}		Integer	1	0 to 255	7+(No. objects *2)…	(List these after all objects have been listed.) Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	8+(No. objects *2)…	Macro ID of the Macro to execute.

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 Tables B.11 and 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 (transport protocol);
- Change Attribute command.

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 Data Mask (to redraw child object or objects).	Change Child Location response
On Change Child Position	Change Child Position command (transport protocol)	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 Attribute	Change Attribute command	For behaviour see other change commands above.	Change Attribute response
On Input Field Selection	Select Input Object 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 Deselection	Select Input Object 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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=5	3	Object type = key
Background colour	1	Integer	1	0 to 255	4	Background colour.
Key code	2	Integer	1	1 to 255	5	Key code assigned by ECU. 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 to 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 to 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.
Repeat: {Object ID}		Integer	2	0 to 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 to 255	8+(No. objects *6)…	(List these after all objects have been listed.) Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	9+(No. objects *6)…	Macro ID of the Macro to execute.

B.7 Button Object

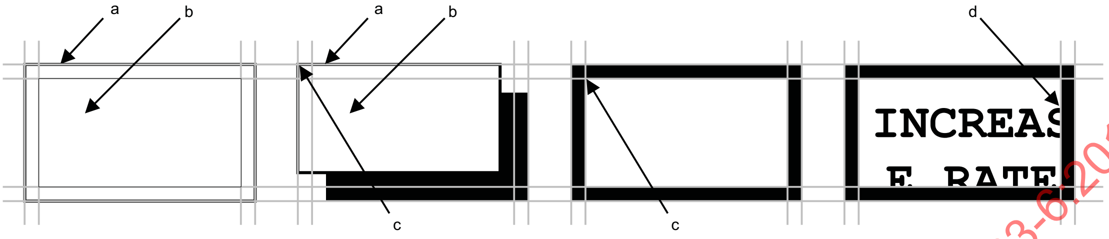
The Button Object defines a button control. This object is intended mainly for VTs with touch screens or a pointing method, but shall be supported by all VTs. Alternatively, if a touch screen or a pointing method is not supported, the VT shall provide a means for navigating to the Button Object or Objects (see 4.6.15). The Working Set can determine if the VT supports touch screen or pointing devices by means of a Get Hardware message. When the Button Object is activated, the VT sends a Button Activation message to the Working Set Master. See Tables B.13 and B.14.

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 may implement child objects. Child objects are clipped to the width and height of the Button Face (see Figure B.1). 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 the area which contains the border colour. Presentation of the border is VT proprietary (see Figure B.1). As a result, the position of the button face is also VT proprietary. The Button Border may be hidden by setting Options – Suppress border bit to TRUE. The Button Border may be eliminated by setting Options – No border bit to TRUE.



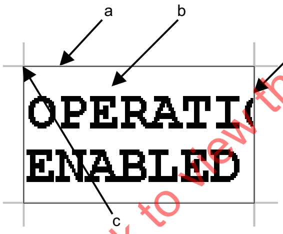
[a] Button Area is determined by width and height attributes.

[b] Button Face is 8 pixels smaller in both width and height.

[c] Button Face offset is VT proprietary (0,0) ⩽ offset ⩽ (8,8).

[d] Objects that do not fit the Button Face are 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 equal to Button Area.

[c] Button Face offset is (0,0).

[d] Objects that do not fit the Button Face are 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 (transport protocol);

— Change Attribute command.

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 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 Deselection	Select Input Object 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	—	Button Activation
On Key Release	Operator releasing the button	—	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 Data Mask (to redraw child object or objects).	Change Child Location response
On Change Child Position	Change Child Position command (transport protocol)	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 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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=6	3	Object Type = Button.
Width	1	Integer	2	0 to 65535	4–5	Maximum width of the button's area in pixels.
Height	2	Integer	2	0 to 65535	6–7	Maximum height of the button's area in pixels.
Background colour	3	Integer	1	0 to 255	8	Background colour.
Border colour	4	Integer	1	0 to 255	9	Border colour.
Key Code	5	Integer	1	0 to 255	10	Key code assigned by ECU. VT reports this code in the Button Activation message.
Options	6[b]	Bitmask	1	[0,1,3] 0 to 63	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. 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.[a] 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.[a] 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.[a] 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.[a] Bits 6 to 7 = 0 and reserved for future use. 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.

Table B.14 (*continued*)

Attribute name	AID	Type	Size bytes	Range or value	Record Byte	Description
Number of objects to follow		Integer	1	0 to 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 to 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.
Repeat: {Object ID}		Integer	2	0 to 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.
{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 to 255	14+(No. objects *6)…	(List these after all objects have been listed.) Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	15+(No. objects *6)…	Macro ID of the Macro to execute.

[a] These bits are present in VT Version 4 and later.

[b] 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, 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:

- hidden (not displayed);
- shown and enabled (displayed and able to accept input);
- shown and disabled (displayed but not able to accept input);

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

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 (excluding Input String object);
- Change String Value command (transport protocol) (Input String object only);
- Change Attribute command;
- Change Size command.

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 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 Deselection	Select Input Object 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 data input, then revert the value of the object to the value before operator began the input (see 4.2). Redraw this object. Refresh parent object.	ESC response (See 4.6.14.)
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 (transport protocol)	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 (transport protocol)
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; nevertheless, it shall fit in the square area specified by the width attribute. An example of a Boolean input is a checkbox. See Figure B.3.

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.

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 shall indicate the state of the Input Boolean to the Working Set with a value in the set {0, 1}. See Table B.16.

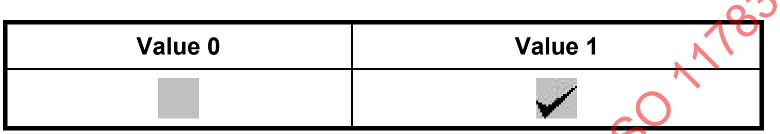


Figure B.3 — Input Boolean examples

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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	= 7	3	Object Type = Input Boolean
Background colour	1	Integer	1	0 to 255	4	Background colour.
Width	2	Integer	2	0 to 65535	5–6	Maximum width and height of the input field in pixels.
Foreground colour	3	Integer	2	0 to 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 to 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 Object ID, the value is stored directly in the value attribute instead.
Value	[5]	Integer	1	0, 1 to 255	11	Value of the input field. 0 for FALSE or > 0 for TRUE. Used only if variable reference attribute is NULL Object ID.
Enabled	[6]	Integer	1	0 or 1	12	Current state of object. 0 = Disabled, 1 = Enabled
Number of macros to follow		Integer	1	0 to 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.
Repeat: {Event ID}		Integer	1	0 to 255	14…	Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	15…	Macro ID of the Macro to execute.

B.8.3 Input String object

This object is used to input a character string from the operator. Displayable characters are shown in Tables L.1 to L.7. Several special formatting characters are permitted in the Input String value and shall be properly interpreted by the VT, as specified (see 4.6.16.5). See Table B.17.

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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=8	3	Object Type = Input String
Width	1	Integer	2	0 to 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 to 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 to 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 to 65534	9–10	Object ID of a Font Attributes object to use for display formatting of this field.
Input attributes	5	Integer	2	0 to 65534, 65535	11–12	Object ID of an Input Attributes object or Extended Input Attributes object to use for character string validation or NULL Object ID for no validation. A referenced Input Attribute must be of the same type as the string value (or String Variable), in that both must 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 extended input object and the Value is a WideString.
Options	6	Bitmask	1	[0 to 3] 0 to 7	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 4.6.16.4). Bit 2 = Wrap on Hyphen. If TRUE, Auto-Wrapping can occur between a hyphen ($2D_{16}$) and the following character (see 4.6.16.4). 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.[a]
Variable reference	7	Integer	2	0 to 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 Object ID, the value is stored directly in the value attribute instead. **IMPORTANT — For Version 3 VTs and prior, if this attribute ⩽ 65534, the Length attribute shall be set to zero and the value attribute *excluded* from this record.**

Table B.17 (*continued*)

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record Byte**	**Description**
Justification	8	Integer	1	[0 to 2] 0 to 15	16	Field justification. Indicates how the text string is positioned within the field defined by width and height (see 4.6.16.1). 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[a] 0 = Position Top 1 = Position Middle 2 = Position Bottom 3 = Reserved
Length		Integer	1	0 to 255	17	Maximum fixed length of the Input String value in bytes. This may be set to zero 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 Value command (see H.18) is only one byte.
Value		String	Length		18...	Value of the input field. Used only if variable reference attribute is NULL Object ID. 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 4.6.16.6). The string type (8 bit/WideString) shall not be changed by the VT; however, the Working Set may cause the type to change from an 8 bit String to a WideString or vice versa.
Enabled	[9]	Integer	1	0 or 1	Depends on size of value attribute	Current state of object. 0 = Disabled, 1 = Enabled
Number of macros to follow		Integer	1	0 to 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.
Repeat: {Event ID}		Integer	1	0 to 255	Depends on size of value attribute	Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	Depends on size of value attribute	Macro ID of the Macro to execute.

[a] VT Version 4 and later.

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:

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

Depending on the “Options” attribute given in Table B.18, displayed values are either truncated or rounded to the number of decimals specified in the “# of decimals” attribute.

The displayed value shall be formatted according to the above equation even if the value is outside the min./max. value range.

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 are true:

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

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

Scaled min value $\leqslant$ new value $\leqslant$ 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:

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

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

See Table B.18.

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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=9	3	Object Type = Input Number
Width	1	Integer	2	0 to 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 to 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 to 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 to 65534	9–10	Object ID of a Font Attributes object to use for display formatting of this field.
Options	5	Bitmask	1	[0 to7] 0 to 15	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. 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 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.[a] [b] The designer should account for a unary minus sign with respect to leading zeros and the field width.
Variable reference	6	Integer	2	0 to 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 Object ID, the value is stored directly in the value attribute instead. VT transmits the raw unscaled value to the Working Set.
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 Object ID. 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.

Table B.18 (*continued*)

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record Byte**	**Description**
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.
# of decimals	11	Integer	1	0 to 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 to 2] 0 to 15	36	Field justification. Indicates how the number is positioned within the field defined by width and height (see 4.6.16.1). 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 During data input of this object, the VT designer may choose to suppress justification until the field is closed after input.
Options 2	[15]	Integer	1	[0, 1] 0 to 3	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 data input.[b] If TRUE, the value shall be transmitted to the ECU as it is being changed (see 4.2).
Number of macros to follow		Integer	1	0 to 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.
Repeat: {Event ID}		Integer	1	0 to 255	39…	Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	40…	Macro ID of the Macro to execute.

[a] Prior to VT Version 4, the behaviour was undefined.

[b] VT Version 4 and later.

B.8.5 Input List object

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 would 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 to 254).

In VT Version 3 and prior, the behaviour with value 255 was not defined. Value 255 is used to indicate that no item is chosen. The CF may 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 shall not be allowed to set the value to an invalid index (an index 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 Object ID or a container, and when the container is in the hidden state, the list item 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 Object ID 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 Object ID);
- selected list item is an Object Pointer with a value of NULL Object ID;
- selected list item is a Container, and the container is in the hidden state.

See Tables B.19 and B.20.

Allowed commands:

- Enable/Disable Object command;
- Select Input Object command;
- ESC command;
- Change Numeric Value;
- Change Attribute command;
- Change List Item command;
- Change Size command.

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 Deselection	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 data input, then revert the value of the object to the value before operator began the input (see 4.2). Redraw this object. Refresh parent object.	ESC response (See 4.6.14.)
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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=10	3	Object Type = Input List
Width	1	Integer	2	0 to 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 to 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 to 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 Object ID, the value is stored directly in the value attribute instead.
Value	[4]	Integer	1	0 to 254, 255	10	Selected list index of this object. Used only if variable reference attribute is NULL Object ID. The current list item chosen, or 255 to indicate no item chosen. The first item is at index zero (0).
Number of list items		Integer	1	0 to 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]	Integer	1	[0, 1] 0 to 3	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 data input.[a] If TRUE, the value shall be transmitted to the ECU as it is being changed (see 4.2).
Number of macros to follow		Integer	1	0 to 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.
Repeat: {object ID}		Integer	2	0 to 65534, 65535	14+object* 2	These objects make up the list. The NULL Object ID 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 to 255	14+(No. List Items * 2)…	(List these after all objects have been listed.) Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	15+(No. List Items * 2)…	Macro ID of the Macro to execute.

[a] 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 different attributes. See Tables B.21 to B.25.

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

- Change Background Colour command;
- Change Numeric Value;
- Change String Value command (transport protocol) (Output String object only);
- Change Attribute command;
- Change Size command.

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 (transport protocol)	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 Tables L.1 to L.7. Several special formatting characters are permitted in the Output String value and shall be properly interpreted by the VT as specified in 4.6.16.5.

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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=11	3	Object Type = Output String
Width	1	Integer	2	0 to 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 to 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 to 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 to 65534	9–10	Object ID of a Font Attributes object to use for display formatting of this field.
Options	5	Bitmask	1	[0 to 3] 0 to 7	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 4.6.16.4). Bit 2 = Wrap on Hyphen. If TRUE, Auto-wrapping can occur between a hyphen ($2D_{16}$) and the following character (see 4.6.16.4). 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.[a]
Variable reference	6	Integer	2	0 to 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 Object ID, the string is stored directly in the value attribute instead. **IMPORTANT — For Version 3 VTs and prior, if this attribute ⩽ 65534, the Length attribute shall be set to zero and the value attribute EXCLUDED from this record.**

Table B.22 (*continued*)

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record Byte**	**Description**
Justification	7	Integer	1	[0 to 2] 0 to 15	14	Field justification. Indicates how the text string is positioned within the field defined by width and height (see 4.6.16.1). 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[a] 0 = Position Top 1 = Position Middle 2 = Position Bottom 3 = Reserved
Length		Integer	2	0 to 65535	15–16	Maximum fixed length of the output string value in bytes. This may be set to zero if a variable reference is used.
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. May also contain formatting codes as described above. The text string can be 8 bit or WideString (see 4.6.16.6). The Working Set may cause the type to change from an 8 bit String to a WideString or vice versa.
Number of macros to follow		Integer	1	0 to 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.
Repeat: {Event ID}		Integer	1	0 to 255	Depends on size of string	These are listed in pairs: event, Macro, event, Macro etc. Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	Depends on size of string	Macro ID of the Macro to execute.

[a] VT Version 4 and later.

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 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 “# 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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=12	3	Object Type = Output Number
Width	1	Integer	2	0 to 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 to 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 to 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 to 65534	9–10	Object ID of a Font Attributes object to use for display formatting of this field.
Options	5	Bitmask	1	[0 to 7] 0 to 15	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. 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.[a,b]
Variable reference	6	Integer	2	0 to 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 Object ID, the value is retrieved directly from the value attribute instead. VT shall scale the value for display.

Table B.23 (*continued*)

Attribute name	AID	Type	Size bytes	Range or value	Record Byte	Description
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 Object ID. 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.
# of decimals	9	Integer	1	0 to 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 to 2] 0 to 15	28	Field justification. Indicates how the number is positioned within the field defined by width and height (see 4.6.16.1). 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 to 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.
Repeat: {Event ID}		Integer	1	0 to 255	30…	Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	31…	Macro ID of the Macro to execute.

[a] Prior to VT Version 4, the behaviour was undefined.

[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 show is determined by a Value attribute or a variable reference.

If the value of the list (or referenced number variable) is neither 255, nor a valid index within the list (an index which is greater than the number of items in the list minus 1), or if the object at the specified index is an invisible object, then the VT shall not render any of the items from the list.

Allowed commands:

- Change Numeric Value;
- Change Attribute command;
- Change List Item command;
- Change Size command.

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 (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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=37	3	Object Type = Output List
Width	1	Integer	2	0 to 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 to 65535	6–7	Maximum height of the output field in pixels. Objects or portions of objects outside the defined area are clipped.
Variable reference	3	Integer	2	0 to 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 Object ID, the value is found directly in the value attribute instead.
Value	[4]	Integer	1	0 to 254, 255	10	Selected list index of this object. Used only if variable reference attribute is NULL Object ID. 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 to 254	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 to 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.
Repeat: {object ID}		Integer	2	0 to 65534, 65535	13+object* 2	These objects make up the list. The NULL Object ID 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 to 255	13+(No. List Items * 2)…	(List these after all objects have been listed.) Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	14+(No. List Items * 2)…	Macro ID of the Macro to execute.

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 are always drawn using a square “paintbrush”, with the actual point being in the upper left corner of the paintbrush. The width of the paintbrush is given by the line width attribute. The endpoint is relative to the X–Y start location attributes in the parent object. See Figure B.4 and Tables B.26 to B.33.

B.10.2 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.

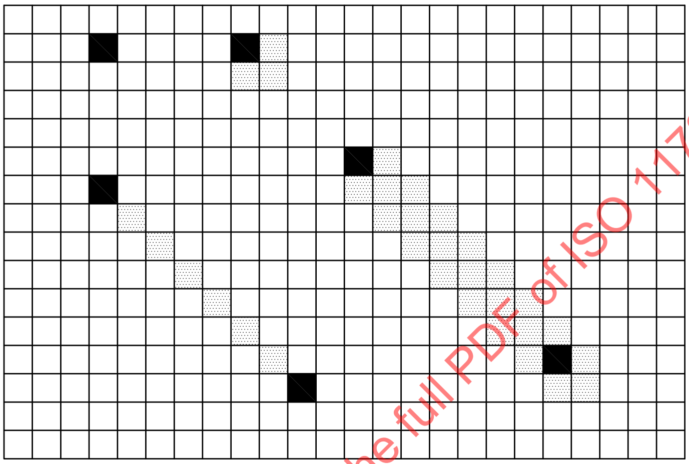


Figure B.4 — Line object showing start and end points using different brush sizes

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 — Line attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record Byte	Description
Object ID		Integer	2	0 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=13	3	Object Type = Line
Line attributes	1	Integer	2	0 to 65534	4–5	Object ID of a Line Attributes object to use for the Line Attributes.
Width	2	Integer	2	0 to 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. Endpoint can be calculated as follows: EndPointX = StartX + Width – LineWidth EndPointY = StartY + Height – LineWidth (See Figure B.4.)
Height	3	Integer	2	0 to 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. Endpoint can be calculated as follows: EndPointX = StartX + Width – LineWidth EndPointY = StartY + Height – LineWidth (See Figure B.4.)
Line Direction	4	Integer	1	0 or 1	10	0 = Line is drawn from top left to bottom right of enclosing virtual rectangle 1 = Line is drawn from bottom left to top right of enclosing virtual rectangle
Number of macros to follow		Integer	1	0 to 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.
Repeat: {Event ID}		Integer	1	0 to 255	12…	Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	13…	Macro ID of the Macro to execute.

B.10.3 Rectangle object

This object outputs a rectangle shape. See Figure 15.

Allowed commands:

- Change Size command;
- Change Attribute command.

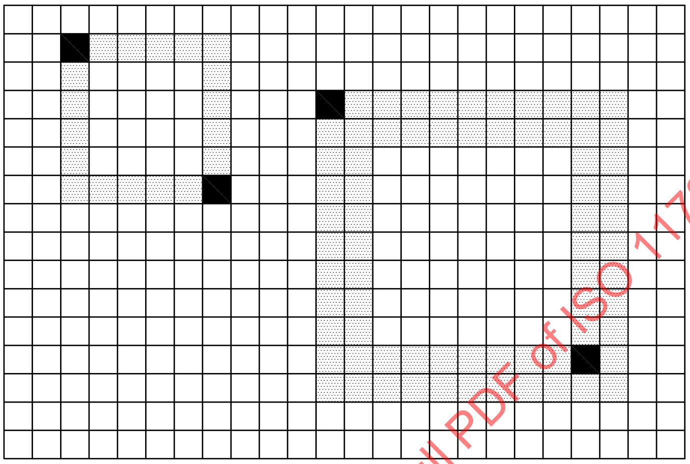


Figure B.5 — Rectangle object showing end points using different brush sizes

Table B.28 — Output Rectangle Events

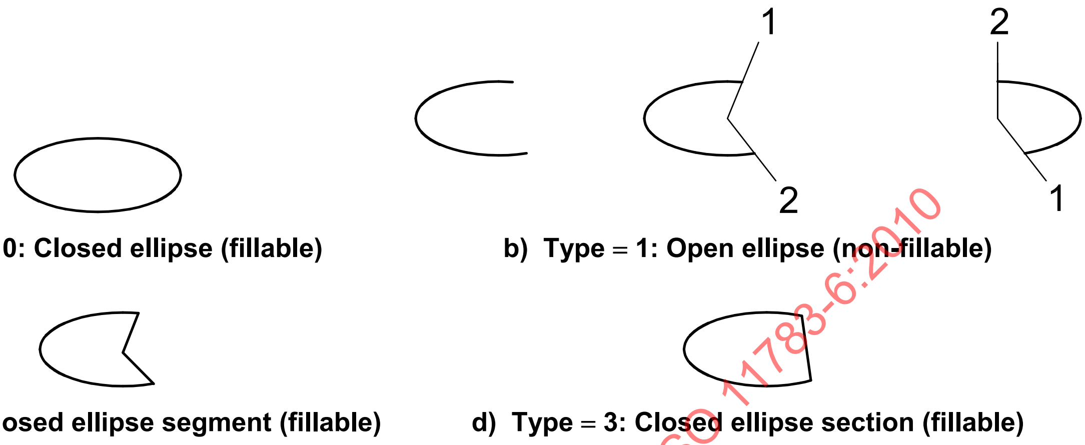
Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Redraw this object.	—
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size response
On Change Attribute	Change Attribute command	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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=14	3	Object Type = Rectangle
Line attributes	1	Integer	2	0 to 65534	4–5	Object ID of a Line Attributes object to use for the Line Attributes.
Width	2	Integer	2	0 to 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. Endpoint can be calculated as follows: EndPointX = StartX + Width – LineWidth EndPointY = StartY + Height – LineWidth See Figure B.5.
Height	3	Integer	2	0 to 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. Endpoint can be calculated as follows: EndPointX = StartX + Width – LineWidth EndPointY = StartY + Height – LineWidth See Figure B.5.
Line suppression	4	Bitmask	1	0 to 15	10	Line suppression. These may 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) 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 B.5. Line width shall be taken into account to know the width of the border.
Fill attributes	5	Integer	2	0 to 65534, 65535	11–12	Object ID of a Fill Attributes object to use for the Fill Attributes or NULL Object ID for no fill.
Number of macros to follow		Integer	1	0 to 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.
Repeat: {Event ID}		Integer	1	0 to 255	14…	Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	15…	Macro ID of the Macro to execute.

B.10.4 Ellipse object

This object outputs an ellipse or circle shape. Several options are available for modifying the appearance (see Figure B.6).



a) Type = 0: Closed ellipse (fillable)

b) Type = 1: Open ellipse (non-fillable)

c) Type = 2: Closed ellipse segment (fillable)

d) Type = 3: Closed ellipse section (fillable)

Key

1 start angle

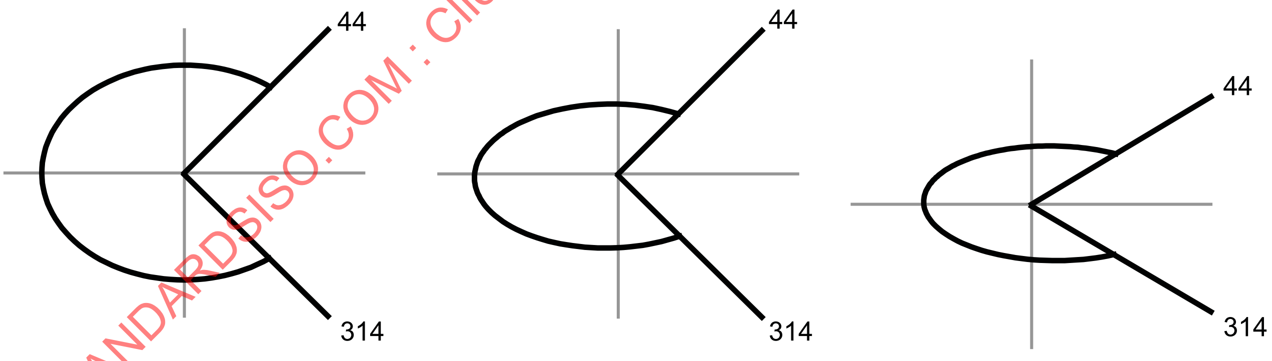
2 end angle

Figure B.6 — Ellipse object

Allowed commands:

- Change Size command;
- Change Attribute command.

Values in degrees



a) Correct

b) Correct

c) Incorrect

Figure B.7 — 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 in order to be accurate. For example, in Figure B.7, 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. 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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=15	3	Object Type = Ellipse
Line attributes	1	Integer	2	0 to 65534	4–5	Object ID of a Line Attributes object to use for the Line Attributes.
Width	2	Integer	2	0 to 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 to 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 to 3	10	Type of ellipse (see Figure B.6): 0 = Closed Ellipse 1 = Open Ellipse defined by start/end angles 2 = Closed Ellipse Segment 3 = Closed Ellipse Section If type > 0 and start and end angles are the same, the ellipse is drawn closed. If type = closed ellipse segment and start and end angles 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 to 180	11	Start angle/2 (in degrees) from positive X axis anticlockwise (90° is straight up). Start and end angles define the arc.
End angle	6	Integer	1	0 to 180	12	End angle/2 (in degrees) from positive X axis anticlockwise (90° is straight up). Start and end angles define the arc.
Fill attributes	7	Integer	2	0 to 65534, 65535	13–14	Object ID of a Fill attributes object to use for the Fill Attributes or NULL Object ID for no fill.

Table B.31 (*continued*)

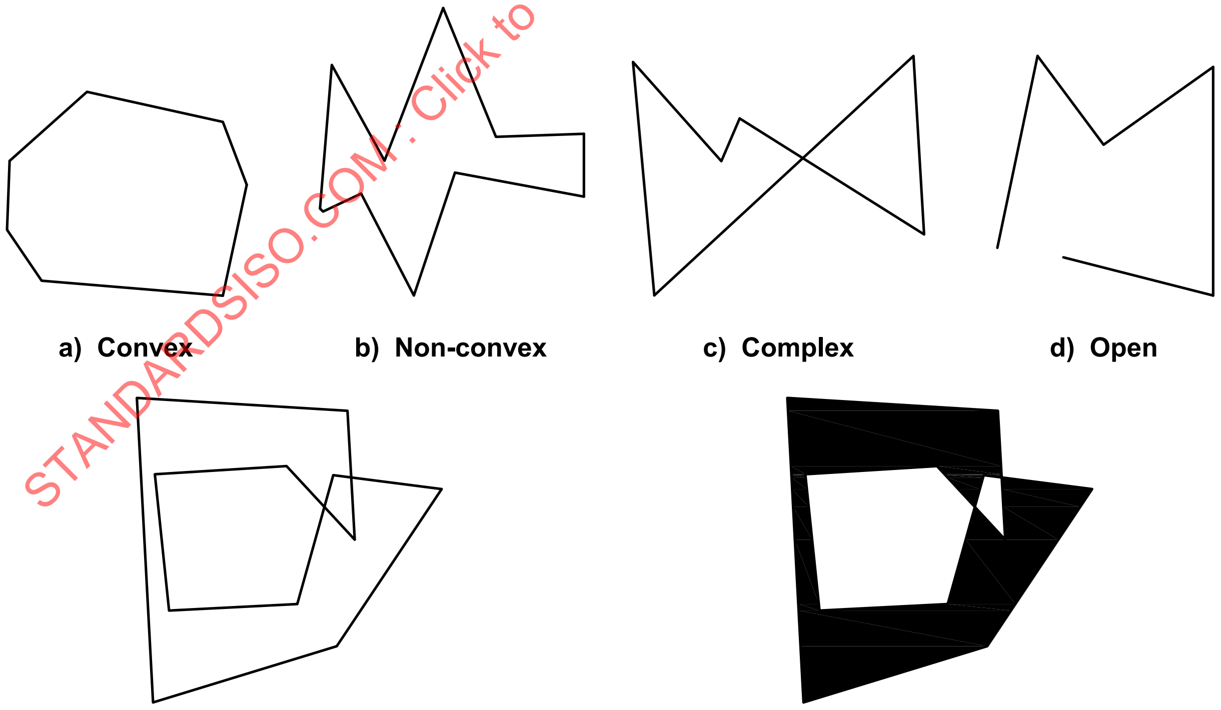
Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record Byte**	**Description**
Number of macros to follow		Integer	1	0 to 255	15	Number of Macro references included even if zero. Each Macro reference consists of 2 bytes: one for event ID and one for Macro ID. Whenever the indicated event occurs, the associated Macro is executed.
Repeat: {Event ID}		Integer	1	0 to 255	16…	Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	17…	Macro ID of the Macro to execute.

B.10.5 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 may also choose to implement only the complex fill algorithm and ignore the polygon type attribute. See Figure B.8.

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 corner of the Polygon object and the upper left corner of the 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.



a) Convex

b) Non-convex

c) Complex

d) Open

e) Polygon unfilled

f) Polygon filled (complex fill algorithm)

Figure B.8 — Polygon types

Allowed commands:

- Change Attribute command;
- Change Size command;
- Change Polygon Point command;
- Change Polygon Scale command.

Table B.32 — Output polygon 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 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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=16	3	Object Type = Polygon
Width	1	Integer	2	0 to 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 to 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 to 65534	8–9	Object ID of a Line Attributes object to use for the Line Attributes.
Fill attributes	4	Integer	2	0 to 65534, 65535	10–11	Object ID of a Fill attributes object to use for the Fill Attributes or NULL Object ID for no fill.
Polygon type	5	Integer	1	0 to 3	12	Polygon type. The first three types are useful only if the polygon is to be filled. VT designer may 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.

Table B.33 (*continued*)

Attribute name	AID	Type	Size bytes	Range or value	Record Byte	Description
Number of points		Integer	1	3 to 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 to 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.
Repeat: {Point X}		Integer	2	0 to 65535	15+point#* 4	X value of a point relative to the top left corner of the polygon.
{Point Y}		Integer	2	0 to 65535	17+point#* 4	Y value of a point relative to the top left corner of the polygon.
Repeat: {Event ID}		Integer	1	0 to 255	15+(num points * 4)…	(List macros after all points.) Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	16+(num points * 4)…	Macro ID of the Macro to execute.

* Indicates multiplication.

B.11 Output graphic objects

B.11.1 General

There are three types of output graphic object: Meter object, Linear Bar Graph object and Arched Bar Graph object.

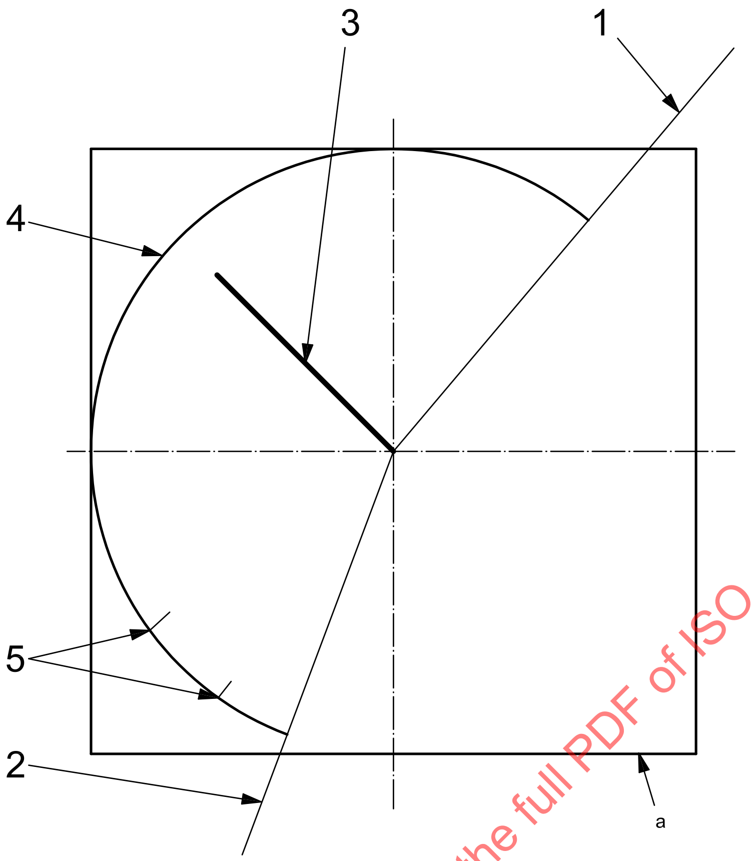
In VT Version 4 and later, if the minimum value is not less than the maximum value, then the object shall be drawn as if the value (or target value) is equal to the minimum value and without regard to the maximum value. Additionally, if the value (or target value) is less than the minimum, the object 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 are not defined.

B.11.2 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 Figures B.9 and B.10, and Tables B.34 and B.35.

It is recommended that the length of any visible tick marks be 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

[a] Meter object cannot exceed boundaries of enclosing square.

Figure B.9 — Meter object

Allowed commands:

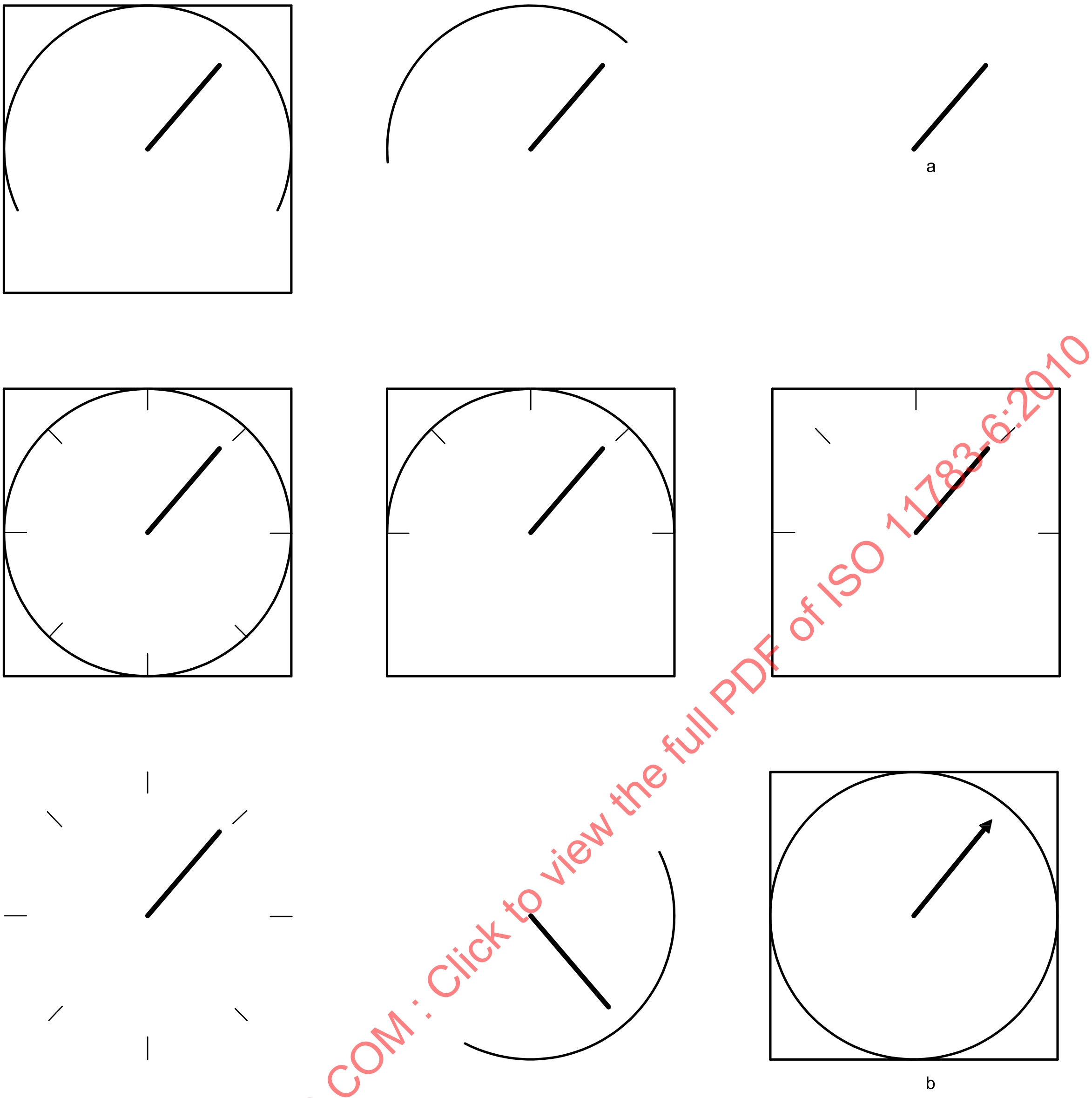
- Change Numeric Value;
- Change Attribute command;
- Change Size command.

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	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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=17	3	Object Type = Output Meter.
Width	1	Integer	2	0 to 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 to 255	6	Needle (indicator) colour.
Border colour	3	Integer	1	0 to 255	7	Border colour (if drawn).
Arc and tick colour	4	Integer	1	0 to 255	8	Meter arc and tick colour (if drawn).
Options	5	Bitmask	1	0 to 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.
# of ticks	6	Integer	1	0 to 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 to 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 to 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 to 65535	13–14	Minimum value. Represents value when needle is at start of arc.
Max. value	10	Integer	2	0 to 65535	15–16	Maximum value. Represents value when needle is at end of arc.
Variable reference	11	Integer	2	0 to 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 Object ID, the value is retrieved directly from the value attribute instead. The referenced Number Variable's value shall be in the range 0 to 65535.
Value	[12]	Integer	2	0 to 65535	19–20	Current value. Needle position is set by this value. Used only if variable reference is NULL Object ID.
Number of macros to follow		Integer	1	0 to 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.
Repeat: {Event ID}		Integer	1	0 to 255	22…	Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	23…	Macro ID of the Macro to execute.



a Needle only.

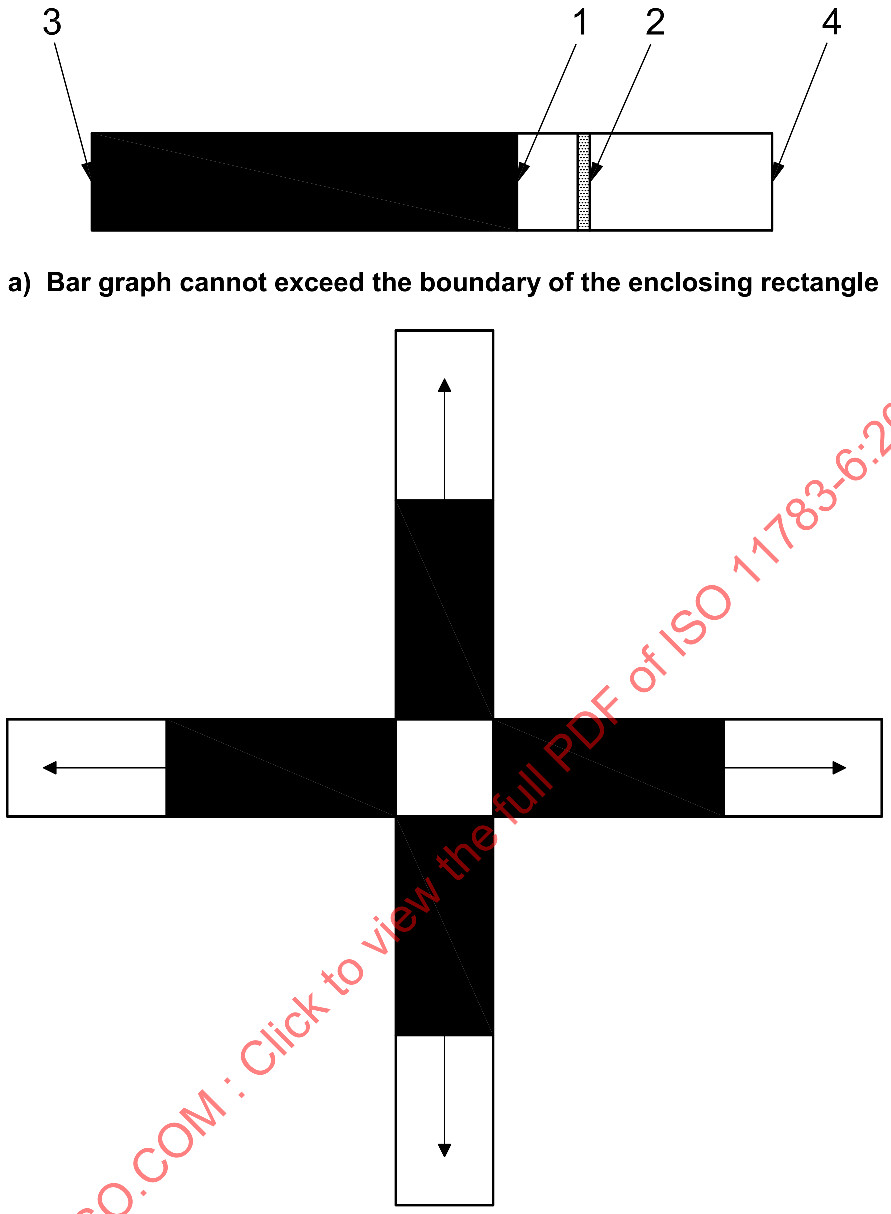
b Appearance of the Meter object is at the discretion of the VT designer.

Figure B.10 — Meter object — Examples

B.11.3 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.

This object is drawn transparent so that objects can be placed underneath to enhance the appearance. See Figure B.11 and Tables B.36 and B.37.



a) Bar graph cannot exceed the boundary of the enclosing rectangle

b) Bar graph can be in any one of four orientations

Key

1 value

2 target value

3 minimum value

4 maximum value

Figure B.11 — Linear Bar Graph — Examples

Allowed commands:

- Change Numeric Value command;
- Change Attribute command;
- Change Size command.

Table B.36 — 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	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 — Linear Bar Graph attributes and record format

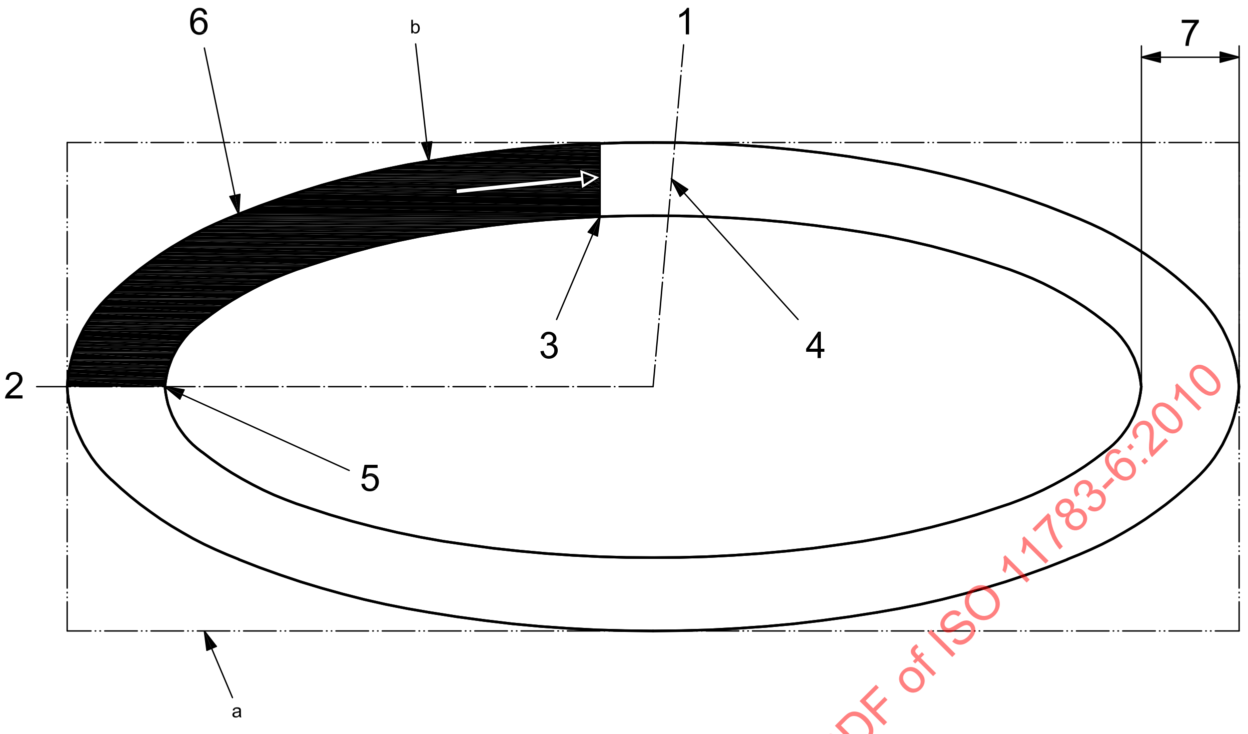
Attribute name	AID	Type	Size bytes	Range or value	Record Byte	Description
Object ID		Integer	2	0 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=18	3	Object Type = Linear Bar Graph
Width	1	Integer	2	0 to 65535	4–5	Maximum width of the enclosing rectangle in pixels. Bar graph cannot exceed the bounds of this imaginary rectangle.
Height	2	Integer	2	0 to 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 to 255	8	Bar graph fill and border colour.
Target line colour	4	Integer	1	0 to 255	9	Target line colour (if drawn).
Options	5	Bitmask	1	0 to 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).
# of ticks	6	Integer	1	0 to 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.

Table B.37 (*continued*)

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record Byte**	**Description**
Min. value	7	Integer	2	0 to 65535	12–13	Minimum value.
Max. value	8	Integer	2	0 to 65535	14–15	Maximum value.
Variable reference	9	Integer	2	0 to 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 Object ID, the value is retrieved directly from the value attribute instead. The referenced Number Variable's value shall be in the range 0 to 65535.
Value	[12]	Integer	2	0 to 65535	18–19	Current value. Used only if variable reference is NULL Object ID. 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 < 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 to 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 Object ID, the target value is retrieved directly from the Target value attribute instead. The referenced Number Variable's value shall be in the range 0 to 65535.
Target value	11	Integer	2	0 to 65535	22–23	Current target value. Used only if Target value variable Reference attribute is NULL Object ID. 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 to 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.
Repeat: {Event ID}		Integer	1	0 to 255	25…	Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	26…	Macro ID of the Macro to execute.

B.11.4 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 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 corner of the enclosing rectangle regardless of orientation. This object is drawn transparent, so that objects can be placed underneath to enhance the appearance. See Figure B.12 and Tables B.38 and B.39.



Key

1	start angle	5	minimum value
2	end angle	6	border
3	value	7	bar graph width
4	maximum value		

[a] Enclosing rectangle, the boundaries of which the bar graph cannot exceed.

[b] In this example, bar graph deflection is clockwise.

Figure B.12 — Arched Bar Graph object — Example

Allowed commands:

- Change Numeric Value;
- Change Attribute command;
- Change Size command.

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	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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=19	3	Object Type = Output Arched Bar Graph.
Width	1	Integer	2	0 to 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 to 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 to 255	8	Bar graph fill and border colour.
Target line colour	4	Integer	1	0 to 255	9	Target line colour (if drawn).
Options	5	Bitmask	1	0 to 31	10	Logical bits to indicate which parts to draw. 1 = TRUE. Bit 0 = Draw border Bit 1 = Draw a target line Bit 2 = Not used 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 to 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 to 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 to 65535	13–14	Bar graph width in pixels. Bar graph width shall be less than half the total width, or less than half the total height, whichever is least (see Figure B.12).
Min. value	9	Integer	2	0 to 65535	15–16	Minimum value.
Max. value	10	Integer	2	0 to 65535	17–18	Maximum value.
Variable reference	11	Integer	2	0 to 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 Object ID, the value is retrieved directly from the value attribute instead. The referenced Number Variable's value shall be in the range 0 to 65535.

Table B.39 (*continued*)

Attribute name	AID	Type	Size bytes	Range or value	Record Byte	Description
Value	[14]	Integer	2	0 to 65535	21–22	Current value. Used only if variable Reference attribute is NULL Object ID. 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 < 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 to 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 Object ID, the target value is retrieved directly from the Target value attribute instead. The referenced Number Variable's value shall be in the range 0 to 65535.
Target value	13	Integer	2	0 to 65535	25–26	Current target value. Used only if target value variable Reference attribute is NULL Object ID. Target value is displayed as a line on the bar graph to indicate some target or warning level. If Target value > Max value or < 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 to 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.
Repeat: {Event ID}		Integer	1	0 to 255	28…	Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	29…	Macro ID of the Macro to execute.

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 Tables B.40 and B.41.

Allowed command:

— Change Attribute command.

Table B.40 — Picture 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

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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=20	3	Object Type = Picture Graphic
Width	1	Integer	2	0 to 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 to 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 to 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 to 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 to 15. 2 = 8 bit colour; 1 colour pixel per byte. Each byte represents a colour palette index of 0 to 255. See Table A.4.
Options	2	Bitmask	1	0 to 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 will be ignored by the VT.)
Transparency colour	3	Integer	1	0 to 255	12	Pixels in the bitmap that have this colour index are transparent (background shows through).
# 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 to 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.

Table B.41 (*continued*)

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record Byte**	**Description**
Repeat: {raw data}		Integer	1	0 to 255	18…	Raw byte of graphic data. Byte 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 is always interpreted left to right, top to bottom of the display. Unused bits at the end of a line are ignored.
{Event ID}		Integer	1	0 to 255	Depends on size of bitmap data	These are listed in pairs: event, Macro, event, Macro, etc. Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	Depends on size of bitmap data	Macro ID of the Macro to execute.

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 Picture Graphic 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. For example, if the size of the Picture Graphic object is 10 pixels wide by two lines high, the format is monochrome and each line is filled with white-coloured pixels. The unused six 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 is 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 is 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 may 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 that Bit 2 of the options attribute be cleared.

The compression algorithm is simple and works as follows. Data is 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, and are never directly included in a mask, key or container. See Table B.42.

Table B.42 — Variable events

Event	Caused by	VT behaviour	Message
On Change Value	Change Numeric Value or Change String Value command (transport protocol)	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.

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 to 65534	1–2	Object identifier. It 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 (transport protocol).

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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=22	3	Object Type = String Variable
Length		Integer	2	0 to 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 4.6.16.6). The Working Set can cause the type to change from an 8 bit String to a WideString 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 never directly included in Mask, Key or Container objects but can be referenced where applicable. There are four types of attribute object: font, line, fill and input. See Tables B.45 and B.46.

B.14.2 Font Attributes object

This object holds attributes related to fonts.

Allowed commands:

- Change Font Attributes command;
- Change Attribute command.

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 and record format

Attribute name	AID	Type	Size bytes	Range or value	Record Byte	Description
Object ID		Integer	2	0 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=23	3	Object Type = Font Attributes
Font colour	1	Integer	1	0 to 255	4	Text colour.
Font size	2	Integer	1	[0 to 14] 0 to 14 or 8 to N	5	Font size If Non-Proportional Font (refer to font style bit 7) Font size value = pixel width x 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 this Font size value of Get Text Font Data response. Width of each character is variable.[c]

Table B.46 (*continued*)

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record Byte**	**Description**
Font type	3	Integer	1	[0, 1, 255] 0 to 255	6	0 = ISO 8859-1 (ISO Latin 1) 1 = ISO 8859-15 (ISO Latin 9) 2 = ISO 8859-2 (ISO Latin 2)[a] 3 = Reserved 4 = ISO 8859-4 (ISO Latin 4)[a] 5 = ISO 8859-5 (Cyrillic)[a] 6 = Reserved 7 = ISO 8859-7 (Greek)[a] 8 – 239 Reserved 240 – 254 = Proprietary[a] 255 = Proprietary If the Font Attributes object applies to a WideString the Font type is ignored (see 4.6.16.6).
Font style	4	Bitmask	1	[0 to127] 0 to 255	7	Font style. These may 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[b] 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).[c]
Number of macros to follow		Integer	1	0 to 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.
Repeat: {Event ID}		Integer	1	0 to 255	9…	These are listed in pairs: event, Macro, event, Macro, etc. Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	10…	Macro ID of the Macro to execute.

[a] For Version 3 and prior VTs, these font types were undefined.

[b] Inverting is to exchange background and pen colours. The rules for background transparency shall be applied.

[c] For Version 4 and later VTs.

B.14.3 Line Attributes object

This object holds Line Attributes related to output shape objects. See Tables B.47 and B.48, and Figure B.13.

Allowed commands:

— Change Line Attributes command;

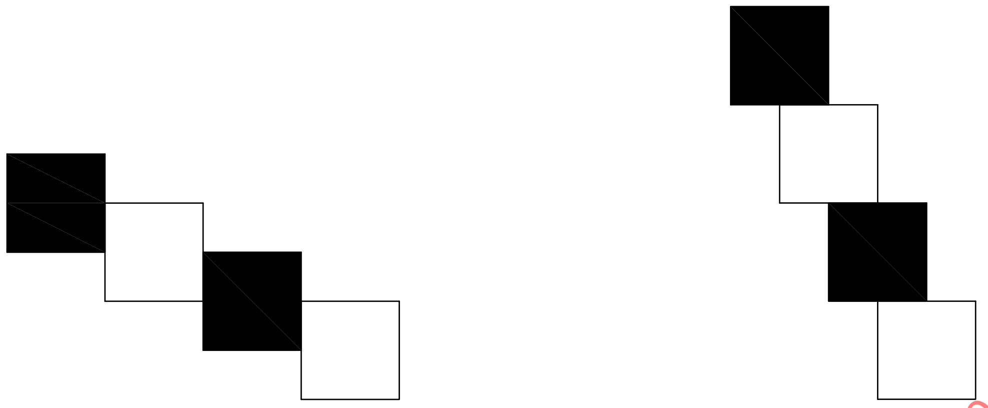
— Change Attribute command.

Table B.47 — Line Attributes events

Event	Caused by	VT behaviour	Message
On Change Line Attributes	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 and record format

Attribute name	AID	Type	Size bytes	Range or value	Record Byte	Description
Object ID		Integer	2	0 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=24	3	Object Type = Line Attributes
Line colour	1	Integer	1	0 to 255	4	Pen colour.
Line width	2	Integer	1	0 to 255	5	Pen thickness in pixels. Lines are drawn with a square paintbrush of this size.
Line art	3	Bitmask	2	0 to 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.13).
Number of macros to follow		Integer	1	0 to 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.
Repeat: {Event ID}		Integer	1	0 to 255	9…	Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	10…	Macro ID of the Macro to execute.



a) More horizontal line

b) More vertical line

Figure B.13 — Effect of Line Attribute — Example pattern: 1010…

B.14.4 Fill Attributes object

This object holds attributes related to filling output shape objects. See Tables B.49 and B.50.

Allowed commands:

- Change Fill Attributes command;
- Change Attribute command.

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 and record format

Attribute name	AID	Type	Size bytes	Range or value	Record Byte	Description
Object ID		Integer	2	0 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=25	3	Object Type = Fill Attributes
Fill type	1	Integer	1	0 to 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 to 255	5	Colour for fill if Fill type = 2. Ignored for all other Fill type values.
Fill pattern	3	Integer	2	0 to 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 NULL Object ID, no fill shall be performed by the VT. **IMPORTANT — In order to simplify monochrome and 16-colour VT design, Picture Graphic objects used as a pattern buffer shall have a width that is byte divisible (i.e. the pattern cannot end somewhere in the middle of a byte).** If this width is not byte divisible, 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).
Number of macros to follow		Integer	1	0 to 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.
Repeat: {Event ID}		Integer	1	0 to 255	9…	These are listed in pairs: event, Macro, event, Macro, etc. Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	10…	Macro ID of the Macro to execute.

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 shall not permit operator entry of invalid characters into the input field. This object is referenced by an Input String object. See Tables B.51 and 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 command:

- Change String Value command (transport protocol).

Table B.51 — Input Attributes events

Event	Caused by	VT behaviour	Message
On Change Value	Change String Value command (transport protocol)	Redraw all objects that are currently displayed and reference this object. Refresh parent object.	Change String Value response

Table B.52 — Input Attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record Byte	Description
Object ID		Integer	2	0 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=26	3	Object Type = Input Attributes
Validation type	[1]	Integer	1	0 to 1	4	0 = valid characters are listed 1 = invalid characters are listed
Length		Integer	1	0 to 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 to 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.
Repeat: {Event ID}		Integer	1	0 to 255	Depends on size of string	These are listed in pairs: event, Macro, event, Macro, etc. Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	Depends on size of string	Macro ID of the Macro to execute.

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 that does not contain a WideString, then no validation shall be performed.

The character ranges defined in this object may 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:

— none.

Table B.53 — Extended Input Attributes and record format

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record Byte**	**Description**
Object ID		Integer	2	0 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=38	3	Object Type = Extended Input Attributes
Validation type	[1]	Integer	1	0 to 1	4	0 = valid characters are listed 1 = invalid characters are listed
Number of code planes to follow		Integer	1	1 to 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 to 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 to 255	7..	Number of character ranges. Each character range consists of two WideChars (first character and last character where the first character $\leqslant$ last character). Depending on validation type (byte 4) the ranges indicate either valid or invalid characters.
Repeat: {{First character}}		Integer	2	0 to 65535	8..	First character in the range.
{{Last character}}		Integer	2	0 to 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

The Object Pointer object allows runtime modification of included objects. By changing the value of Object Pointer object, a different object can be referenced to the same location. The type of object that the Object Pointer object is allowed to point to is limited and depends on the parent object. (Refer to valid parent objects of the Object Pointer object for a list of objects that can be pointed to.) An Object Pointer object can always point to another Object Pointer object. An Object Pointer object can point to the NULL object ID, in which case nothing shall be drawn. See Tables B.54 and B.55.

Allowed command:

— Change Numeric Value.

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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=27	3	Object Type = Object Pointer object
Value	[1]	Integer	2	0 to 65534, 65535	4–5	Object ID of a referenced object or 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). A Macro is defined by a series of one or more command packets. Macro object IDs shall be 255 or less.

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 command:

— none.

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 to 255	1–2	Object identifier. It shall be unique within the object pool. NOTE Although a standard object header is used, the Macro object ID shall be 255 or less.
Type	[0]	Integer	1	=28	3	Object Type = Macro
# of bytes to follow		Integer	2	0 to 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 (transport protocol) on a two byte string], the remaining bytes shall be set to FF_{16} to pad the packet to 8 bytes.
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.

B.17 Colour Map object

The Colour Map object, optionally available in VT Version 4 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 whereby the colour table can be changed at runtime. A Working Set, in using a few colour objects for various backgrounds and borders, can then easily alter the presentation.

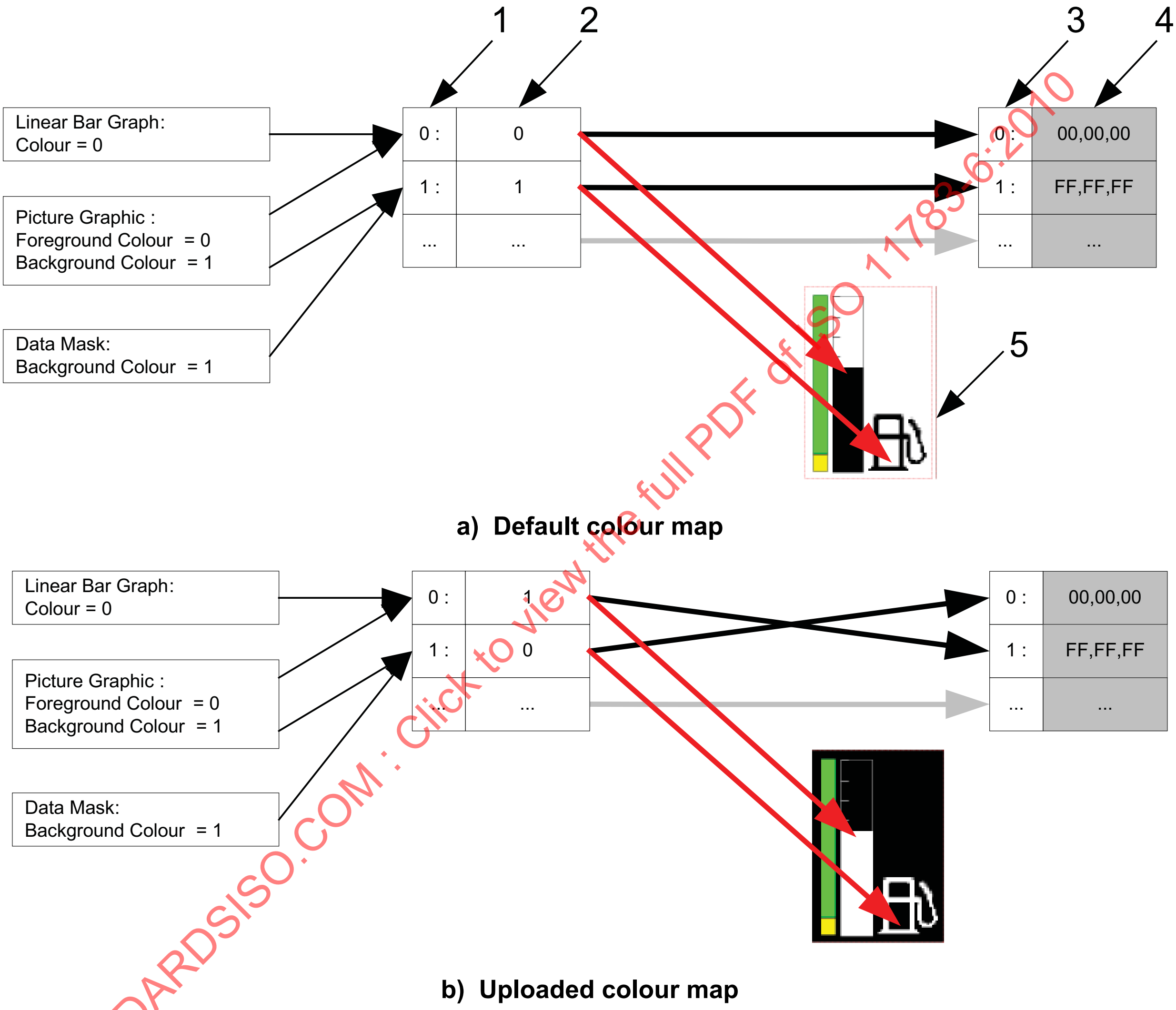
The object pool may contain more than one Colour Map object. After the pool is loaded, the VT uses the default Colour Map. Upon receipt of a valid Select Colour Map command, the VT changes the active palette for this Working Set.

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 VT's capabilities, or the VT will indicate the failure in the Select Colour Map response message. Therefore, a monochrome VT maintains two entries in the Colour Map object, while a 256 colour VT maintains 256 entries in the Colour Map object. The Colour Map object may have capabilities beyond the VT (e.g. a 256 entry Colour Map object may be uploaded for a two-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 B.14 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:

— none.



a) Default colour map

b) Uploaded colour map

Key

1	VT colour index
2	colour map
3	palette index
4	R,G,B value
5	example graphic

Figure B.14 — Colour Map object reverses colours — Example

Table B.57 — Colour Map object attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record Byte	Description
Object ID		Integer	2	0 to 65534	1–2	Object identifier. It 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 to 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, is a bitmap with a canvas and a viewport that can be manipulated by the Working Set at runtime. 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 runtime drawing to the VT screen. This would be useful for precision farming applications, where, for example, the designer could draw a swath 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 runtime.

The contents of this object are never stored by the Store Version command. Working Sets may 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 (see Figure B.15). 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.15.

Instructions	Graph result

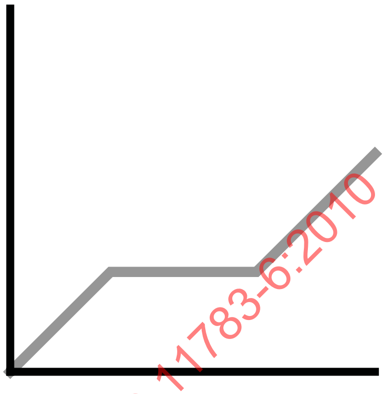
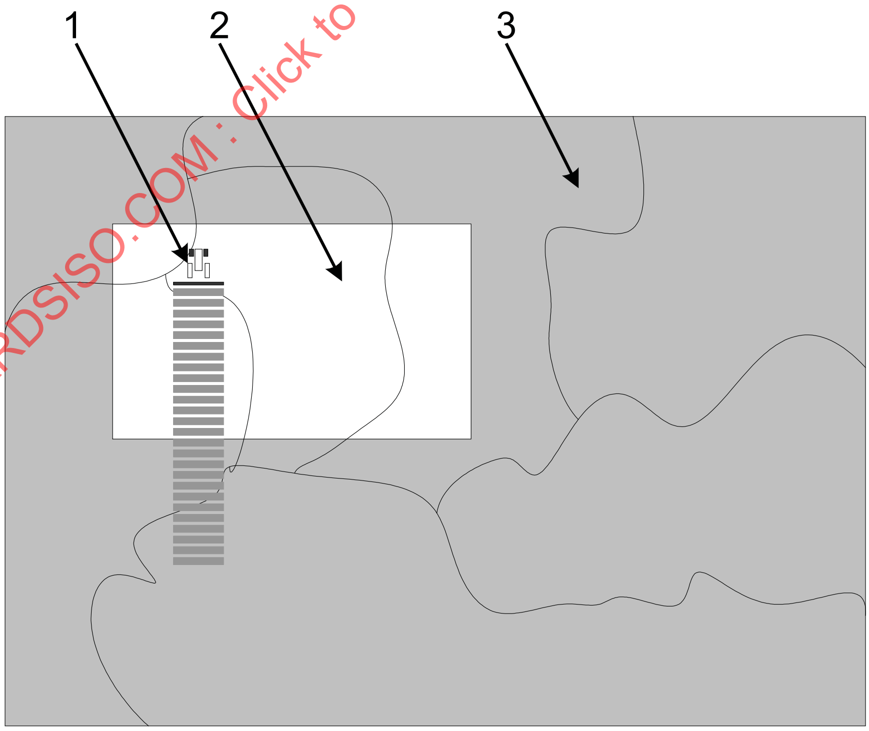
SET FOREGROUND COLOUR (Colour=0 (black))
SET BACKGROUND COLOUR (Colour=1 (white))
SET LINE ATTRIBUTES (object id 6019)
SET FILL ATTRIBUTES (NULL OBJECT ID)
SET GRAPHICS CURSOR (X=0, Y=0)
ERASE RECTANGLE (width=30, height=30)
SET GRAPHICS CURSOR (X=0, Y=0)
DRAW LINE (Xoff=0, Yoff=20)
DRAW LINE (Xoff=20, 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=0, Y=22)
SET FONT ATTRIBUTES (object id 2000)
DRAW TEXT (Transparent, 13, “Graph Example”)
	

Figure B.15 — 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, swath lines could be reset without affected the underlying map image. However, Working Set designers are 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.

See Tables B.58 and B.59.



Key

1 picture graphic object

2 viewport

3 Graphics Context object (entire graphics context referred to as *canvas*)

Figure B.16 — Example application of the Graphics Context object and viewport

Allowed commands:

— Graphics Context command (transport protocol);

— Change Attribute command.

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 to 65534	1–2	Object identifier. It 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 to 32767	4–5	Width of the visible viewport in pixels (on the Data Mask).
Viewport Height	2	Integer	2	0 to 32767	6–7	Height of the visible viewport in pixels (on the Data Mask).
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 to 32767	12–13	Width of the canvas in pixels.
Canvas Height	[6]	Integer	2	0 to 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 to 255	24	Foreground colour to use during drawing when options bit 1 is TRUE.

Table B.59 (*continued*)

Attribute Name	AID	Type	Size bytes	Range or Value	Record Byte	Description
Background Colour	11	Integer	1	0 to 255	25	Background colour to use during drawing when options bit 1 is TRUE. At parsing time, this object is filled with this background colour. Runtime changes to this attribute shall fill this object, effectively erasing any content.
Font Attributes Object	12	Integer	2	0 to 65534, 65535	26–27	Object ID of a Font Attributes Object to use for drawing text. Can be set to NULL Object ID if text is not being used.
Line Attributes Object	13	Integer	2	0 to 65534, 65535	28–29	Object ID of a Line Attributes Object to use for drawing lines and borders or NULL Object ID for line suppression.
Fill Attributes Object	14	Integer	2	0 to 65534, 65535	30–31	Object ID of a Fill Attributes Object to use for filling objects or NULL Object ID for no filling.
Format	15	Integer	1	0 to 2	32	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 to 15. 2 = 8 bit colour; 1 colour pixel per byte. Each byte represents a colour palette index of 0 to 255. See Table A.4.
Options	16	Bitmask	1	0 to 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 zero
Transparency Colour	17	Integer	1	0 to 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

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 4.7.1.1. See Tables B.60 and B.61.

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 (two columns by six 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.

Working Set designers should make the Window Mask object transparent. This allows the VT to set the background colour of the entire User-Layout Data Mask so that all Window Mask objects have the same apparent background colour. If required, the Working Set can determine the VT's background colour using the Get Window Mask Data message.

Allowed commands:

- Change Background Colour command;
- Change Child Location command;
- Change Child Position command (transport protocol);
- Change Attribute command.

Table B.60 — Window Mask events

Event	Caused by	VT behaviour	Message
On Show	A User-Layout Data Mask containing this Window Mask object is displayed on screen	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	A User-Layout Data Mask containing this Window Mask object is removed from the screen		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).	Child Location response
On Change Child Position	Change Child Position command (transport protocol)	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	No action.	Change Attribute response

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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=34	3	Object Type = Window Mask
Width		Integer	1	1 to 2	4	Width (# of User-Layout Data Mask columns). Used only if Window Type attribute is 0 (Free Form).
Height		Integer	1	1 to 6	5	Height (# of User-Layout Data Mask rows). Used only if Window Type attribute is 0 (Free Form).
Window Type		Integer	1	0 to 18	6	Window Type (see B.19.1). 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 8 = 1 × 1 Single Button 9 = 1 × 1 Double Button 10 = 2 × 1 Numeric Output Value with units 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 17 = 2 × 1 Single Button 18 = 2 × 1 Double Button
Background Colour	1	Integer	1	0 to 255	7	Background colour.
Options	2	Integer	1	0 to 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 zero (0).

Table B.61 (*continued*)

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record Byte**	**Description**
Name	3	Integer	2	0 to 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 may 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.
Window Title		Integer	2	0 to 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 NULL Object ID. For window types above zero, the VT may choose to ignore colour and font information and do its own formatting of the text.
Window Icon		Integer	2	0 to 65534, 65535	13–14	Object ID of an output object (as specified in Table A.2) that contains an icon for the window. The VT may use this when formatting window types above type zero and may also use it in the proprietary mapping screen to represent the window. This attribute shall only be NULL Object ID 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 to 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 zero. For all other window types, refer to the tables that follow in B.19.1. Each attribute that follows is a two-byte reference to an object ID or NULL Object ID.
Number of objects to follow		Integer	1	0 to 255	16	Number of objects to follow even if zero. If the Window Type attribute is not zero, this attribute shall be set to zero. 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.
Number of macros to follow		Integer	1	0 to 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.

Table B.61 (*continued*)

Attribute name	AID	Type	Size bytes	Range or value	Record Byte	Description
Repeat: {Object ID}		Integer	2	0 to 65535	18+object*2	Object ID of a required object reference for the given window type (see B.19.1). List all objects before listing macros.
Repeat: {Object ID}		Integer	2	0 to 65534	18 + num referenced objects*2	Object ID of an object contained in this object (see A.1.3). List all objects before listing macros.
{X Location}		Signed integer	2	−32768 to +32767	20 + num referenced objects*2	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	22 + num referenced objects*2	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 to 255	18 + num referenced objects*2 + num child objects*6	(List these after all objects have been listed.) Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	19 + num referenced objects*2 + num child objects*6	Macro ID of the Macro to execute.

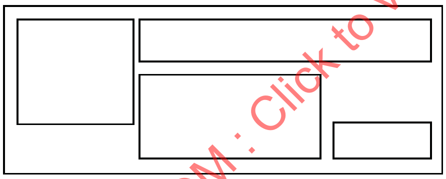
B.19.1 Window Mask Window Type

The Window Mask Window 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 Window Mask Window Type is zero (0 = Free Form) the Working Set may create a completely customized presentation.

B.19.1.1 Free Form Window (0)

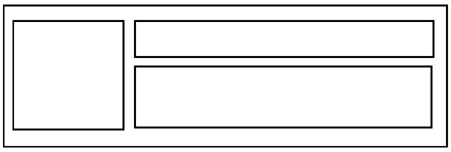
When the Window Mask Window Type attribute is 0, the Working Set supplies and positions all child objects contained inside the window. In this case the Working Set has complete control over the look and feel of the window and the VT shall render the Window Mask exactly as specified by the child objects, position, size and transparency attributes of the Window Mask object, and all other formatting attributes.

B.19.1.2 1 × 1 Numeric Output Value Window With Units

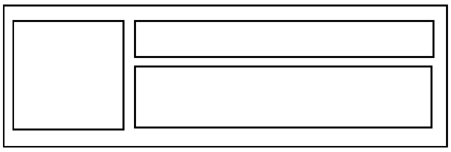
Window Type	1
Description	This window displays a single numeric output with units of measure in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric value and units of measure
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 may format the window as desired, and may ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT may 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 4.7.15.2.
Example Layout	 This is an example that shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.3 1 × 1 Numeric Output Value Window, No Units

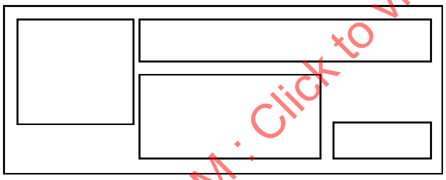
Window Type	2
Description	This window displays a single numeric output with no units of measure in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric value
Number of Object 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 may format the window as desired, and may ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT may 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 4.7.15.2.
Example Layout	 This is an example that shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.4 1 × 1 String Output Value Window

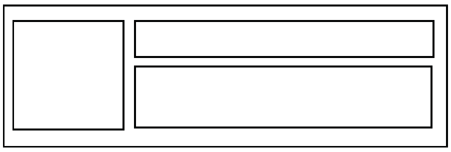
Window Type	3
Description	This window displays a single string output in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	String Value
Number of Object References	1
Object Reference	#1 — Output String (for the string value)
VT Field Lengths	Window Title: 11 characters Window Value: 11 characters
VT Formatting and Scaling	The VT may format the window as desired, and may ignore colour and Font Attributes in the referenced object. Field length shall conform to the above requirements. The VT may 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 4.7.15.2.
Example Layout	 This is an example that shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.5 1 × 1 Numeric Input Value Window With Units

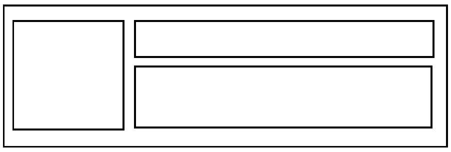
Window Type	4
Description	This window displays a single numeric input with units of measure in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric value and units of measure
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 may format the window as desired, and may ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT may 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 4.7.15.2.
Example Layout	 This is an example that shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.6 1 × 1 Numeric Input Value Window, No Units

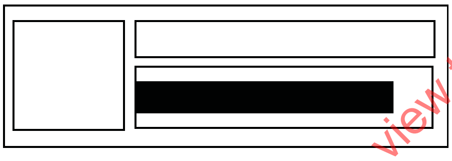
Window Type	5
Description	This window displays a single numeric input with no units of measure in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric value
Number of Object 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 may format the window as desired, and may ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT may 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 4.7.15.2.
Example Layout	 This is an example that shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.7 1 × 1 String Input Value Window

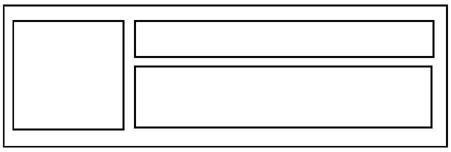
Window Type	6
Description	This window displays a single string input in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	String Value
Number of Object References	1
Object Reference	#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 may format the window as desired, and may ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT may 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 4.7.15.2.
Example Layout	 This is an example that shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.8 1 × 1 Horizontal Linear Bar graph Window

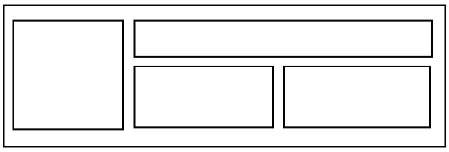
Window Type	7
Description	This window displays a single horizontal linear bar graph in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Bar Graph
Number of Object References	1
Object Reference	#1 — Linear Bar Graph
VT Field Lengths	Window Title: 11 characters
VT Formatting and Scaling	The VT may format the window as desired, and may ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT may 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 4.7.15.2. The Working Set shall supply a Linear Bar Graph object in the horizontal position. The bar graph should increase from left to right but may increase in either direction.
Example Layout	 This is an example that shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.9 1 × 1 Single Button Window

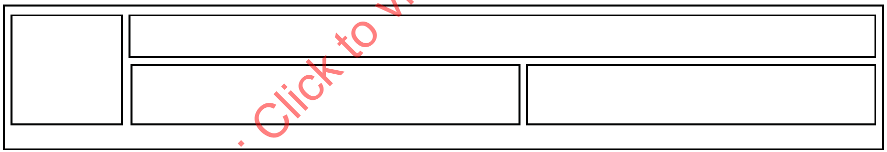
Window Type	8
Description	This window displays a single Button Object in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	One Button
Number of Object References	1
Object Reference	#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 may 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 4.7.15.2. 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 is an example that shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.10 1 × 1 Double Button Window

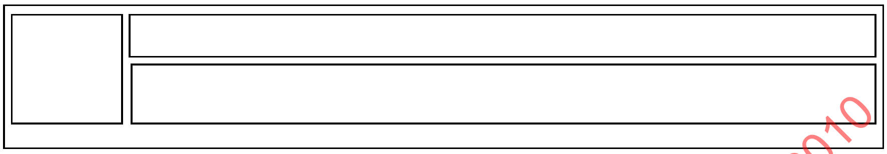
Window Type	9
Description	This window displays two Button Objects in a single window cell.
Window Designator	Window Icon, Window Title. While both are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Two Buttons
Number of Object References	2
Object References (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 may 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 4.7.15.2. The Working Set shall also scale the Button Objects and their children according to these equations: Button Width = Window Cell Width × 30 % (rounded down) Button Height = Window Cell Height × 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 is an example that shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.11 2 × 1 Numeric Output Value Window With Units

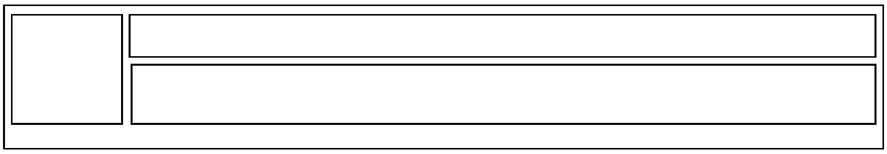
Window Type	10
Description	This window displays a single numeric output with units of measure in two horizontal window cells.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric value 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 may format the window as desired, and may ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT may 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 4.7.15.2.
Example Layout	 This is an example only and shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.12 2 × 1 Numeric Output Value Window, No Units

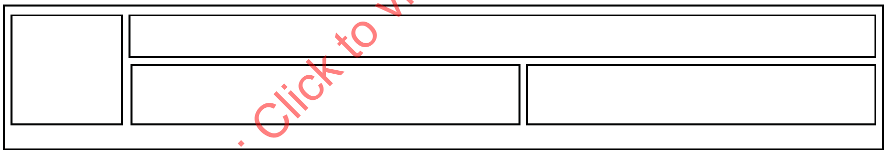
Window Type	11
Description	This window displays a single numeric output with no units of measure in a single window cell.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric value
Number 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 may format the window as desired, and may ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT may 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 4.7.15.2.
Example Layout	 This is an example only and shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.13 2 × 1 String Output Value Window

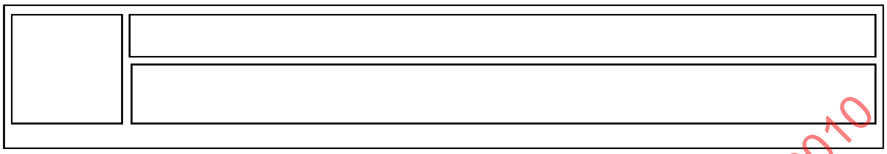
Window Type	12
Description	This window displays a single string output in a single window cell.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	String Value
Number of Object References	1
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 may format the window as desired, and may ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT may 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 4.7.15.2.
Example Layout	 This is an example only and shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.14 2 × 1 Numeric Input Value Window With Units

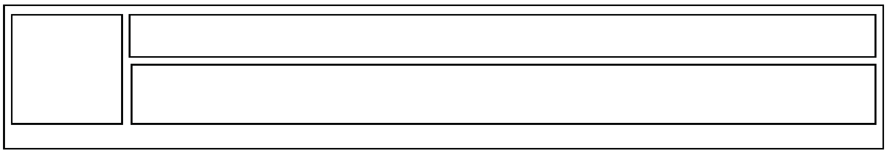
Window Type	13
Description	This window displays a single numeric input with units of measure in a single window cell.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric value and units 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 may format the window as desired, and may ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT may 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 4.7.15.2.
Example Layout	 This is an example only and shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.15 2 × 1 Numeric Input Value Window, No Units

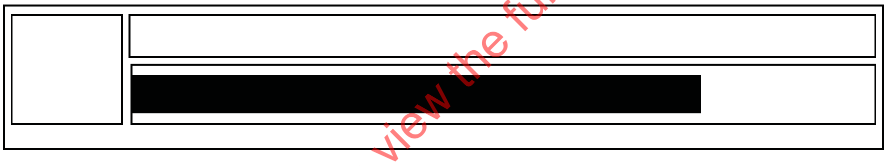
Window Type	14
Description	This window displays a single numeric input with no units of measure in a single window cell.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Numeric Value
Number 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 may format the window as desired, and may ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT may 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 4.7.15.2.
Example Layout	 This is an example only and shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.16 2 × 1 String Input Value Window

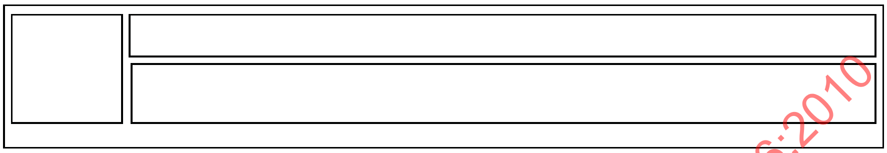
Window Type	15
Description	This window displays a single string input in a single window cell.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	String Value
Window Value Type	Input String 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 may format the window as desired, and may ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT may 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 4.7.15.2.
Example Layout	 This is an example only and shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.17 2 × 1 Horizontal Linear Bar graph Window

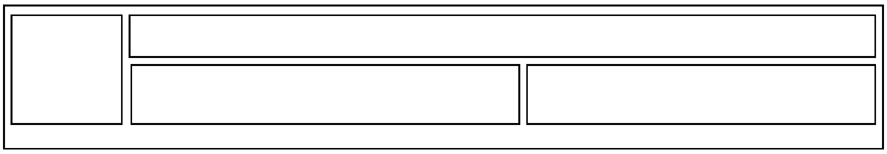
Window Type	16
Description	This window displays a single horizontal linear bar graph in a single window cell.
Window Designator	Window Icon, Window Title. While both together are optional, at least one of these items shall be displayed by the VT.
Required to be Displayed	Bar Value
Number of Object References	1
Object Reference	#1 — Linear Bar Graph
VT Field Lengths	Window Title: 20 characters
VT Formatting and Scaling	The VT may format the window as desired, and may ignore colour and Font Attributes in the referenced objects. Field length shall conform to the above requirements. The VT may 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 4.7.15.2. The Working Set shall supply a linear bar graph object in the horizontal position. The bar graph may grow in either direction but left to right is recommended.
Example Layout	 This is an example only and shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.18 2 × 1 Single Button Window

Window Type	17
Description	This window displays a single Button Object in a single window cell.
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 may 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 4.7.15.2. 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 is an example only and shows all components of the window. VT designs control the formatting and layout of this window.

B.19.1.19 2 × 1 Double Button Window

Window Type	18
Description	This window displays two Button Objects in a single window cell.
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 may 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 4.7.15.2. 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 NULL Object ID. See Tables B.62 and B.63.

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 NULL Object ID reserve a Key position (the remaining Keys do not move up).

Pointers to 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.

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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=35	3	Object type = Key Group
Options	1	Integer	1	0 to 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 zero (0).
Name	2	Integer	2	0 to 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 may 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.
Key Group Icon		Integer	2	0 to 65534, 65535	7–8	Object ID of an output object (as specified in Table A.2) that contains an optional icon for the key group. The VT may use this in the proprietary mapping screen to represent the key group. It is recommended that a transparent background not be used 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 to 4	9	Number of Key or Object Pointer objects to follow. After dereferencing pointers, there shall be a maximum of four Key Objects per Key Group object.
Number of macros to follow		Integer	1	0 to 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.
Repeat: {Object ID}		Integer	2	0 to 65534	11+object*2	Object ID of a Key Object contained in this Key Group (see A.1.3). List all objects before listing macros.
Repeat: {Event ID}		Integer	1	0 to 255	11+(No. objects *2)…	(List these after all objects have been listed.) Event ID of event type that causes this Macro to execute.
{Macro ID}		Integer	1	0 to 255	12+(No. objects *2)…	Macro ID of the Macro to execute.

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. See Table B.64.

An object label is only intended for use by the VT in various proprietary screens and popup messages or editors, and is 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 may be used by the VT to identify the Working Set in proprietary screens and alarms (e.g. a communication alarm).
- It is recommended that Working Set designs provide an Object Label for all input objects. Since popup editor windows in the VT may 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 (metres)” and/or an appropriate designator graphic.

Strings in excess of 32 characters may be clipped to 32 characters by the VT. Referenced designator graphics shall fit within a Soft Key designator area (see 4.5.2). The referenced designator object may 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.

Allowed commands:

- Change Object Label command.

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 to 65534	1–2	Object identifier. It shall be unique within the object pool.
Type	[0]	Integer	1	=40	3	Object Type = Object Label Reference List
Number of Labelled objects	[1]	Integer	2	0 to 65535	4–5	Number of labelled objects to follow. One labelled object consumes 7 bytes.
Repeat: {Object ID}		Integer	2	0 to 65534	6–7..	Object ID of object to label.
{String Variable reference}		Integer	2	0 to 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 to 255	10..	Font type (See Annex L) (ignored if String Variable object reference is NULL Object ID or the string contains a WideString (See 4.6.16.6).
{Graphical reference}		Integer	2	0 to 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.

Annex C
(normative)

Object transport protocol

C.1 Virtual terminal messages and object transfer

Two PGNs are reserved for the VT message protocol, as follows.

a) VT to ECU

Transmission repetition rate:	As required
Data length:	8 bytes
Data page field:	0
PDU format field:	230
PDU specific field:	Destination address
Default priority:	7
Parameter group number:	58880 ($00E600_{16}$)

b) ECU to VT

Transmission repetition rate:	As required
Data length:	8 bytes
Data page field:	0
PDU format field:	231
PDU specific field:	Destination address
Default priority:	7
Parameter group number:	59136 ($00E700_{16}$)

Before a Working Set Master builds an object pool in a VT, it may 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.

Annex E defines messages for storing an object pool in a VT non-volatile memory for loading into the VT's operating read/write memory on power up. This transfer can be via PC Cards, or diagnostic or programming tools.

A Working Set's object pool shall be transferred into the VT's memory by some means. This could be by data card, hard-storage in the VT's firmware or indirectly via an ISO 11783 network.

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 the transport protocol specified in ISO 11783-3 and the extended transport protocol specified in Annex K. Destination-specific messages shall be used and connection management shall be implemented. Extended transport protocol is required to permit transfers of up to 117 Mbytes.

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 shall be used (see Annex K). 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 as 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. Sending an invalid pool with objects or macros that do not parse 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.

C.2.2 Object pool transfer procedure

The following procedure is 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 may use this request to allocate memory for the pool. The Working Set Master shall check the error codes returned. If no error is reported, the Working Set Master may proceed.

b) The Working Set Master uses the transport protocol or extended transport protocol or both (see Annex K) to move the object pool to the VT using the object pool transfer message (see C.2.3). Normal handshaking, error checking and retransmission, 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.

c) The following governs the transfer of the object pool:

    1) The Working Set Master may send several TP or ETP sessions or both 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 may 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.

    3) Transfer sessions containing no object records are not permitted.

d) Upon completion, 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.

e) 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.

f) After sending the End of Object Pool message, the Working Set Master shall wait for an End of Object Pool response. 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 may 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 4.6.6. 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. At that time, if the End of Object Pool response message has not been received by the Working Set Master, then it shall assume that the End of Object Pool message was not received by the VT. Three messages are waited for, to avoid race conditions created by a VT Status message that could already be in a transmit queue and which does not correctly identify the parsing state.

g) Commands are sent from a Working Set to the VT using the PGNs specified in this annex. Only Working Set Masters (not Members) are allowed to send any of the commands specified in this annex. The originating master shall wait for a response before sending another of these commands.

C.2.3 Object pool transfer message (transport protocol)

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)	

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, transmit as 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.

As a VT can take a long time to parse a pool, the End of Object Pool response shall be delayed until this activity is completed. 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, transmitted as 0 (zero) Bit 4 = 1 = any other error Bit 5-7 = Reserved, transmitted as 0 (zero)	
Bytes	3, 4		Parent Object ID of faulty object, transmit as NULL object ID if there are no object pool errors	
Bytes	5, 6		Object ID of faulty object, transmit as 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, transmitted as 0 (zero)	
Byte	8		Reserved, transmit as FF_{16}	

C.2.6 Updating pools at runtime

If the Working Set needs to modify an object such that the length of the object will change (e.g. changing the length of a string, adding/deleting macros or child objects), then the Working Set shall update its pool at runtime. New objects can also be added to the object pool with this procedure. 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 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. If no error is reported, the Working Set Master may proceed.

b) The Working Set Master uses extended transport protocol, or transport protocol, to move the object or objects to the VT. Normal handshaking, error checking and retransmission, in accordance with Annex K and ISO 11783-3, shall be implemented [see C.2.2 b)].

c) Upon completion, 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.

d) The VT responds with the End of Object Pool response (see C.2.5).

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.

A pool update can take several seconds or even minutes and if, for example, the operator presses a Soft Key while the pool update is running, then the VT shall execute macros triggered and commands sent from that Working Set.

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 the 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 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 commands given in Annex F whenever 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 the Working Set and the response by the VT. These messages are not part of the object pools and are not allowed in macros. Working Set masters or members may send any command specified in this annex.

The VT shall respond to these commands even if the requesting ECU is not part of a Working Set.

D.2 Get Memory message

The Get Memory message allows the Working Set to determine if the VT is out of memory and also determines the VT version.

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
Byte	2			Reserved, transmit as FF_{16}	
Bytes	3–6	Memory Required		Number of bytes in the object pool	
Bytes	7, 8			Reserved, transmit as FF_{16}	

In Version 3 and prior VTs, the “Memory Required” parameter represents the number of bytes in the object pool to be transferred. In Version 4 and later VTs, the “memory required” parameter shall be calculated using the sum of the number of bytes in the object pool to be transferred and the estimated storage of all Graphics Context objects. The storage space required for a single Graphics Context object shall be estimated as follows:

SIZE = ((Width in pixels / bit depth) [rounded up to nearest byte]) * height in pixels

Where:

Bit depth monochrome VT = 8

Bit depth 16 colour VT = 4

Bit depth 256 colour VT = 1

For more details on the use of the Memory Required parameter, see C.2.2 and C.2.6.

The Working Set should send the Get Memory message with Memory Required = 0 in order to receive the response indicating the VT version number. This information may be used to calculate the actual size of the pool to be transferred. The VT should not allocate any storage for this request.

D.3 Get Memory response

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 edition of ISO 11783-6 that this VT meets 0 = Hannover Agritechnica 2001 limited feature set 1 = The version of the FDIS (Final Draft International Standard) 2 = IS Version ISO 11783-6:2004(E), First Edition, 2004-06-15 3 = IS Version ISO 11783-6:2010(E), Second Edition, [ISO 11783-6:2004(E) and features specifically noted with Version 3 reference] 4 = IS Version ISO 11783-6:2010(E), Second Edition (this document in its entirety)	
Byte	3			Status 0 = There can be enough memory.[1] 1 = There is not enough memory available. Do not transmit Object Pool.	
Bytes	4–8			Reserved, transmit as FF_{16}	

D.4 Get Number of Soft Keys message

The Get Number of Soft Keys message supplies the Working Set with 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.

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, transmit as 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

[1] Because there is overhead associated with object storage, it is impossible to predict whether there is enough memory available.

Byte	2	Navigation Soft Keys	Version 3 and Prior: Reserved, transmit as FF_{16} 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, transmit as 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 provides the Working Set with 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 Text Font Data
Bytes	2–8			Reserved, transmit as 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, transmit as FF_{16}	
Byte	6	Small font sizes			(Values are width by 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	Not used
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	Not used

Byte	8	Type attribute		Supported 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 supported (Version 4 and later VTs)

D.8 Get Hardware message

The Get Hardware message informs the Working Set as to 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, transmit as 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		Maximum number of seconds from a power cycle to transmission of first “VT Status message” (see G.2). Transmit as FF_{16} when this information is not available.[2]	
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 to 15 and monochrome and 16 colour Picture Graphic objects). 2 = 256 Colour (VT supports colour codes 0 to 255 and all formats of Picture Graphic objects).	

[2] VT Version 4 and later.

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 4.6.15)[3)] Bit 5 = 1 = VT supports simultaneous activations of all combinations of Buttons (see 4.6.15)[3)] Bit 6 = 1 = VT reports drag operation via Pointing Event message (Bit 0 or Bit 1 shall be set to 1)[3)] Bit 7 = 1 = VT supports intermediate coordinates during a drag operation (Bit 6 shall be set to 1)[3)]
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.

D.10 Get Supported Widechars message

This message only applies to Version 4 and later VTs.

The Get Supported Widechars message supplies the Working Set with a list of the WideChars supported by the VT.

The message only requests characters from a single code plane. If the ECU 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 $\leqslant$ Last WideChar.

The ECU 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}–$1FFFF_{16}$ etc.
Bytes	3, 4			First WideChar in inquiry range	
Bytes	5, 6			Last WideChar in inquiry range	
Bytes	7, 8			Reserved, transmit as FF_{16}	

[3)] These bits exist in VT Version 4 and later.

D.11 Get Supported WideChars response (transport protocol)

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 00000_{16}–$0FFFF_{16}$ 1 => characters 10000_{16}–$1FFFF_{16}$ etc.
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) Bits 1-3 = Reserved, transmitted as 0 (zero) Bit 4 = 1 = any other error Bits 5-7 = Reserved, transmitted as 0 (zero)	
Byte	8	Number of ranges		Indicates the number of entries in the WideChar range array. Set to zero 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 ECU does not have to request this message because the VT shall display WideStrings even if they contain unsupported characters (see 4.6.16.6). The VT shall include the characters from the WideChar minimum character set when responding to a Code Plane 0 request (see Table L.7).

EXAMPLE Response from a VT supporting only the WideChar Minimum Character Set. The ECU 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.

The Get Window Mask Data message provides the Working Set with 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, transmit as FF_{16}	

D.13 Get Window Mask Data response

This message only applies to Version 4 and later VTs.

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	0101	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, transmit as FF_{16}	

D.14 Get Supported Objects message

This message is available in VT Version 4 and later. This command is used by the WS to get the list of all object types supported by the VT.

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 = 197_{10}			
	Bits	7–4	1100	Command	Get Technical Data
	Bits	3–0	0101	Parameter	Get Supported Objects
Bytes	2–8			Reserved, transmit as 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.

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 objects types that are optional for the VT to support;
- proprietary object types that the VT supports.

The VT may send the list of proprietary object types to the WS. The VT and WS may decide by another means which proprietary objects may be supported.

In either case, whether or not the VT lists the proprietary objects in this message, the VT may still decide to reject the object pool from the WS because it included proprietary objects.

The WS may 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 and Auxiliary Function Type 1 objects in the list of supported objects.

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 may choose to use the same object pool in both cases.

Transmission repetition rate:	In response to Get Supported Objects message
Data length:	Variable
Parameter group number:	VT to ECU, destination-specific
Allowed in a Macro:	No

Byte 1	VT function = 197_{10}			
	Bits 7–4	1100	Command	Get Technical Data
	Bits 3–0	0101	Parameter Response	Get Supported Objects
Byte 2			Number of bytes to follow (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. The special value FF_{16} is reserved for future use, but if found by the WS during parsing, indicates the end of the list, and all following bytes should be ignored.	

Annex E
(normative)

Non-volatile memory operations commands

E.1 General

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 Load Version command 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. Depending on the VT design, either only a single object pool or an arbitrary number of pools can be managed for each Working Set. If more than one pool per Working Set is managed by the VT, each pool should be identified by a seven-character-wide version label.

Version labels may be displayed to an operator and may 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 L.1). Version labels shall be padded with trailing blanks to produce a seven-character string. 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 its 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 specified 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 specified in this annex.

Because non-volatile operations can take an indefinite amount of time to complete, the response message could be delayed accordingly. Therefore, the VT Status message shall reflect 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 following messages are not allowed in macros.

E.2 Get Versions message

The Get Versions message allows the Working Set to query the VT for existing 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, transmit as FF_{16}	

E.3 Get Versions response (transport protocol)

The VT sends all version labels contained in the non-volatile storage associated with the requesting Working Set. If there is no version stored, the number of version strings is set to 0. Transport protocol is used when needed.

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 version in non-volatile storage shall be overwritten; alternatively, if there is no 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 are filled up with blanks. 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, transmit as FF_{16}	
Byte	6				Acknowledgment	
	Bits	7–4		0000	Not used	
	Bits	3–0		0000	Successfully stored	
				0001	Not used	
				0010	Version label is not correct	
				0100	Insufficient memory available	
				1000	General error	
Bytes	7, 8				Reserved, transmit as 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 may proceed as if all objects had been transmitted normally. If the version label contains no string (all blanks), the last stored 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 could already be in a transmit queue and 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 are filled up with blanks. 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, transmit as FF_{16}	
Byte	6			Acknowledgment[4]	
	Bits	7–4	0000	Not used	
	Bits	3–0	0000	Successfully loaded	
			0001	Unsuccessfully loaded, file system error or pool data corruption	
			0010	Version label is not correct or Version label unknown	
			0100	Insufficient memory available	
			1000	General error	
Bytes	7, 8			Reserved, transmit as FF_{16}	

E.8 Delete Version command

The Delete Version command allows a Working Set to delete a 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 version in non-volatile storage is to be deleted. For deleting 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 are filled up with blanks. Only 8 bit Strings are allowed.	

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, transmit as FF_{16}	
Byte	6			Acknowledgment	
	Bits	7–4	0000	Not used	
	Bits	3–0	0000	Successfully deleted	
			0001	Not used	
			0010	Version label is not correct or Version label unknown	
			0100	NA	
			1000	General error	
Bytes	7, 8			Reserved, transmit as FF_{16}	

[4] Byte 6, 0=1 is a requirement for VT Version 4 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 may send any command specified in this annex. The originating master/member shall wait for a response up to a maximum of 1,5 s before sending another command, unless stated otherwise. Each of the commands in this annex can also be used in a Macro unless otherwise noted. Working Set designers should recognize that the VT can take a significant amount of time to respond to a command, especially if the command causes a display refresh (refer to the busy codes in the VT Status message, in G.2).

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. If the object cannot be displayed due to references to missing objects, the VT generates an error in the response.

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, transmit as 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 = object is Hidden, 1 = object is Shown	
Byte	5				Error Codes (0 = no errors)	
					Bit 0 = 1 = References to missing objects	
					Bit 1 = 1 = Invalid object ID	
					Bit 2 = 1 = Command error	
					Bit 3 = not used	
					Bit 4 = 1 = Any other error	
Bytes	6–8				Reserved, transmit as FF_{16}	

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.

It is allowed to enable already-enabled objects and to disable already-disabled objects, in which cases the response frame shall indicate “no errors” and macros shall be executed.

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				0 = Disable, 1 = Enable	
Bytes	5–8				Reserved, transmit as 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 = 161_{10}				
		Bits	7–4	1010	Command	Command
		Bits	3–0	0001	Parameter	Enable/Disable Object
Bytes	2, 3				Object ID	
Byte	4				0 = object is Disabled, 1 = object is Enabled	

Byte	5	Error Codes (0 = no errors) Bit 0 = 1 = Not used Bit 1 = 1 = Invalid object ID Bit 2 = 1 = Command error Bit 3 = 1 = Could not complete. Operator input is active on this object. Bit 4 = 1 = Any other error
Bytes	6–8	Reserved, transmit as 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).

If the object to be selected is included multiple times on the same mask, it is proprietary to the VT as to which of the object instances will be selected (has focus).

Even if the input field is activated for data input, the value shall not be changed by this command (e.g. Input Boolean is not toggled).

NOTE 1 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 2 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 Object ID indicates that no object shall be selected (i.e. focus is removed)	
Byte	4				Option FF_{16} = Set Focus to object referenced by Object ID 0 = Activate for data input object referenced by Object ID (invalid for Button Object or Key Object) NOTE Value 0 available only on VT Version 4 and later.	
Bytes	5–8				Reserved, transmit as FF_{16}	

F.7 Select Input Object response

The VT uses this message to respond to the Select Input Object command.

Transmission repetition rate:	In response to a Select Input Object command
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 NULL object ID 1 = Object referenced by Object ID is Selected 2 = Object referenced by Object ID is Opened for Edit[5]
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[5] 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, transmit as FF$_{16}$

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, transmit as 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 = Not used Bit 4 = 1 = Any other error	
Bytes	5–8			Reserved, transmit as FF$_{16}$	

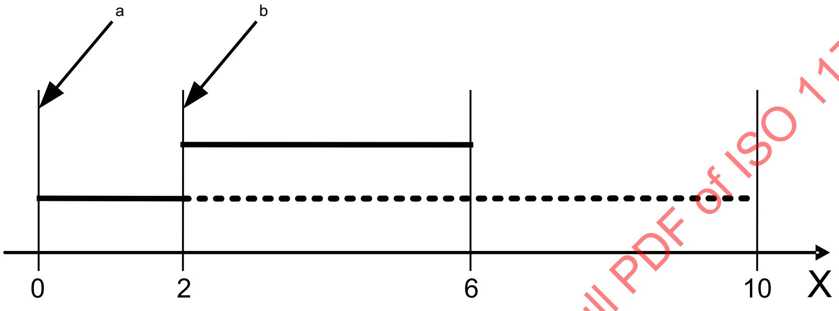
[5] These bits/values exist in VT Version 4 and later.

F.10 Control Audio Signal command

This command may be used to control the audio on the VT. When received, this message shall terminate any audio in process from the originating ECU 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 could 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 produce 278 min of continuous tone. If this is insufficient, the originating ECU may 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 ECU at a time, then the Control Audio Signal response shall indicate that the Audio Device is busy. The audio produced by a control audio command shall never be queued or delayed beyond normal VT message processing. See Figure F.1.



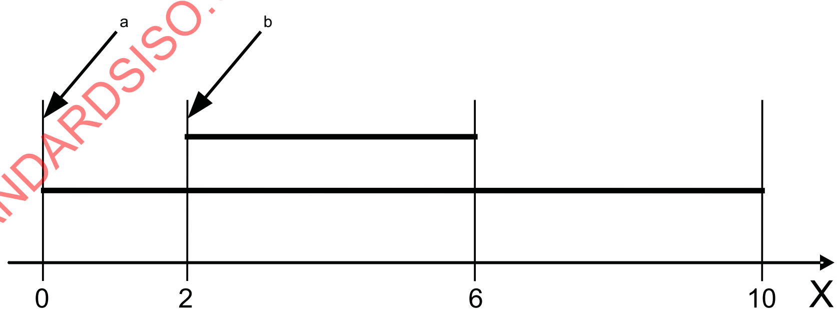
[a] Working Set 1 sends Control Audio Device command with a total time of 10 s.

[b] 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

If the VT is capable of supporting the Control Audio Signal command for more than a single ECU at a time, then the audio for each Working Set is not interrupted. See Figure F.2.

NOTE The Control Audio Signal command is independent of the currently active Working Set, therefore audio tones can be commanded whether or not the originating ECU is the active Working Set (see 4.6.11).



[a] Working Set 1 sends Control Audio Device command with a total time of 10 s.

[b] 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

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 ECU (Frequency and Duration values are ignored). 1–255 = Number of Audio Activations.	
Bytes	3, 4		Frequency in Hz. If the Frequency specified is outside 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 less than the VT capabilities for timing, the VT shall time the audio to the VT's 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 VT's smallest controlled value	

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 = not used Bit 4 = 1 = Any other error	
Bytes	3–8		Reserved, transmit as 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.

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	
Bytes	3–8			Reserved, transmit as 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 Bits 1–3 = not used Bit 4 = 1 = Any other error	
Bytes	3–8			Reserved, transmit as FF_{16}	

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). 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).

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, transmit as 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 = Invalid Parent Object ID Bit 1 = 1 = Invalid Object ID Bits 2–3 = Not used, send as 0 Bit 4 = 1 = Any other error	
Bytes	7, 8			Reserved, transmit as FF_{16}	

F.16 Change Child Position command (transport protocol)

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 (use Change Child Location command instead). When the object is moved, the parent object shall be refreshed. The position attributes given in the message are signed integers. 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 (transport protocol).

Transmission repetition rate:	In response to Change Child Position command (transport protocol)
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 = Invalid Parent Object ID Bit 1 = 1 = Invalid Object ID Bits 2–3 = Not used Bit 4 = 1 = Any other error	
Bytes	7, 8			Reserved, transmit as 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, transmit as 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	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 = not used Bit 2 = not used Bit 3 = not used Bit 4 = 1 = Any other error	
Bytes	5–8			Reserved, transmit as FF_{16}	

F.20 Change Background Colour command

This command is used to change the background colour of an object.

NOTE Version 4 and later VTs support a means of transforming the colour table into alternate mapping (see B.17).

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, transmit as 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 = not used Bit 4 = 1 = Any other error	
Bytes	6–8			Reserved, transmit as 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
Bytes	2, 3			Object ID of object to change	
Byte	4			Reserved, transmit as 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.

Boolean input object:	1 byte for TRUE/FALSE
Number input object:	4 bytes for integer input
List input object:	1 byte for list index
List output object:	1 byte for list index
Number output object:	4 bytes for integer output
Meter:	2 bytes for integer value
Linear bar graph:	2 bytes for integer value
Arched bar graph:	2 bytes for integer value
Number variable:	4 bytes for integer value
Object pointer:	2 bytes for Object ID

The frequency of update is at the discretion of the Working Set designer, but overloading the VT should be avoided. Values should be updated only when they change.

F.23 Change Numeric Value response

The VT sends this message in response to the Change Numeric Value.

Transmission repetition rate:	In response to Change Numeric Value
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
Bytes	2, 3		Object ID	
Byte	4		Error Codes (0 = no error) Bit 0 = 1 = Invalid object ID Bit 1 = 1 = Invalid value Bit 2 = 1 = Value in use (e.g. open for input)[6] Bit 3 = not used 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):	

Boolean input object:	1 byte for TRUE/FALSE
Number input object:	4 bytes for integer input
List input object:	1 byte for list index
List output object:	1 byte for list index
Number output object:	4 bytes for integer output
Meter:	2 bytes for integer value
Linear bar graph:	2 bytes for integer value
Arched bar graph:	2 bytes for integer value
Number variable:	4 bytes for integer value
Object pointer:	2 bytes for Object ID

[6] VT Version 4 and later.

F.24 Change String Value command (transport protocol)

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. The Working Set shall complete the entire string transfer before beginning another string transfer session.

If the message contents fit in a single packet, transport protocol shall not be used; however, the packet shall be padded with data to, and including, Byte 8. Any data needed to fill a packet that is beyond the total number of bytes in the string shall be ignored.

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:	Depends on string size
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, transmit as FF_{16}	
Bytes	4, 5			Object ID of the object to change	
Byte	6			Error Codes (0 = no error) Bit 0 = 1 = Not used Bit 1 = 1 = Invalid Object ID Bit 2 = 1 = String too long Bit 3 = 1 = Any other error Bit 4 = 1 = Value in use (e.g. open for input)[7]	
Bytes	7, 8			Reserved, transmit as FF_{16}	

[7] VT Version 4 and later.

F.26 Change End Point command

This command is used to change the end point of a 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 a Line object to change	
Bytes	4, 5			Width in pixels	
Bytes	6, 7			Height in pixels	
Byte	8			Line Direction (refer to 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 = not used	
				Bit 3 = not used	
				Bit 4 = 1 = Any other error	
Bytes	5–8			Reserved, transmit as FF_{16}	

F.28 Change Font Attributes command

This command is used to change the Font Attributes in a Font Attributes object.

NOTE Version 4 and later VTs support a means of transforming the colour table into alternate mapping (see B.17).

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, transmit as 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, transmit as FF_{16}	

F.30 Change Line Attributes command

This command is used to change the Line Attributes in a Line Attributes object.

NOTE Version 4 and later VTs support a means of transforming the colour table into alternate mapping (see B.17).

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	
Byte	8			Reserved, transmit as FF_{16}	

F.31 Change Line Attributes response

The VT uses this message to respond to the Line Attributes command.

Transmission repetition rate:	In response to the 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 = not used	
				Bit 4 = 1 = Any other error	
Bytes	5–8			Reserved, transmit as FF_{16}	

F.32 Change Fill Attributes command

This command is used to change the Fill Attributes in a Fill Attributes object.

NOTE Version 4 and later VTs support a means of transforming the colour table into alternate mapping (see B.17).

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, transmit as 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, transmit as 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 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	
Byte	4, 5			New Active Mask Object ID	
Bytes	6–8			Reserved, transmit as 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
Byte	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 = 1 = Not used Bit 3 = 1 = Not used Bit 4 = 1 = Any other error	
Bytes	5–8			Reserved, transmit as FF_{16}	

F.36 Change Soft Key Mask command

This command is used to change the Soft Key Mask associated with a data or 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	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, transmit as FF_{16}	

F.37 Change Soft Key Mask response

The VT uses this message to respond to the Change Soft Key Mask message (see H.16).

Transmission repetition rate:	In response to Change Soft Key Mask
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			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, transmit as 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	Change Attribute
Bytes	2, 3			Object ID of object to change	
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):	
				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 = Value in use (e.g. open for input)[8] Bit 4 = 1 = Any other error
Bytes	6–8	Reserved, transmit as FF_{16}

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 either becomes the active Working Set and mask or is no longer 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, transmit as FF_{16}	

F.41 Change Priority response

The VT uses this message to respond to the 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 Mask	
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 = not used Bit 4 = 1 = Any other error	
Bytes	6–8			Reserved, transmit as FF_{16}	

[8] VT Version 4 and later.

F.42 Change List Item command

This command is used to change a list item in an Input List object or Output List object. Output List object 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 = 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 or Output List 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, transmit as 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	0001	Parameter	Change List Item
Bytes	2, 3			Object ID of an Input List object or Output List object[9]	
Byte	4			List Index (items are numbered 0–n)	
Bytes	5, 6			New Object ID or $FFFF_{16}$ to set empty item	
Byte	7			Error Codes (0 = no error) Bit 0 = 1 = Invalid Input List object ID or Output List object ID[9] Bit 1 = 1 = Invalid List Index Bit 2 = 1 = Invalid New List Item Object ID Bit 3 = 1 = Value in use (e.g. open for input)[9] Bit 4 = 1 = Any other error	
Byte	8			Reserved, transmit as FF_{16}	

[9] VT Version 4 and later.

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

NOTE For deleting 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, transmit as 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 = not used Bit 4 = 1 = Any other error	
Bytes	3–8			Reserved, transmit as FF_{16}	

F.46 Lock/Unlock Mask command

This message is available in VT Version 4 and later. This command is used by an ECU 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 will be used when a series of changes needs 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 following unlock mechanisms occurs:

a) an Unlock command is received and the visible mask has been refreshed;

b) a timeout occurs based on the timeout attribute of the Lock message;

c) navigation to, or activation of, input objects or buttons on the Data Mask;

d) a change of visible mask occurs;

e) the pool is deleted;

f) a proprietary reason (e.g. an input dialog closes).

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 Window Mask is visible. If a Data Mask or Window Mask is not visible (e.g. the 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 Window Mask is visible, the VT shall not respond to the Unlock Mask command until the Data Mask or window mask has been completely refreshed (all changes during lock made visible to the operator). If a timeout occurs or a change in visible Data Mask or Window Mask occurs, the VT shall send an unsolicited Lock/Unlock Mask Response message with appropriate error codes set.

Typically, the Working Set will 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.

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 Window Mask 1 = Lock Data Mask or Window Mask	
Bytes	3, 4			Object ID of the Data Mask or Window Mask to Lock. 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, transmit as FF_{16}	

F.47 Lock/Unlock Mask response

This message is available in VT Version 4 and later. The VT uses this message to respond to the Lock/Unlock Mask command or to send an unsolicited message on timeout or visible mask change.

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 Window Mask 1 = Lock Data Mask or Window 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, the active mask changed Bit 6 = 1 = Unsolicited unlock, operator induced, or any other error Bit 7 = 1 = Any other error
Bytes	4–8	Reserved, transmit as FF_{16}

F.48 Execute Macro command

This message is available in VT Version 4 and later. This command is used to execute a Macro.

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, transmit as FF_{16}	

F.49 Execute Macro response

This message is available in VT Version 4 and later. The VT uses this message to respond to the Execute Macro command.

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, transmit as FF_{16}	

F.50 Change Object Label command

This message is available in VT Version 4 and later. This command is used by an ECU to change a label of an object.

Transmission repetition rate:	On request
Data length:	8
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 L) (ignored if String Variable object reference is NULL Object ID or the string contains a WideString (see 4.6.16.6).	
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 available in VT Version 4 and later. This message is sent in response to the Change Object Label command.

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 Bit 1 = 1 = Invalid String Variable object id Bit 2 = 1 = Invalid font type Bit 3 = 1 = Any other error Bit 4 = 1 = Designator references invalid objects Bit 5 = 1 = Any other error	
Bytes	3–8			Reserved, transmit as FF_{16}	

F.52 Change Polygon Point command

This message is available in VT Version 4 and later. This command is used by a Working Set to modify a point in a Polygon 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 = 182_{10}			
	Bits	7–4	1011	Command	Command
	Bits	3–0	0110	Parameter	Change Polygon Point
Bytes	2, 3			Object ID of the 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

This message is available in VT Version 4 and later. The VT uses this message to respond to the Change Polygon Point command.

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 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, transmit as FF_{16}	

F.54 Change Polygon Scale command

This message is available in VT Version 4 and later. This command is used by a Working Set to change the scale of a complete Polygon object.

This message causes the value of the polygon points to be changed. It is similar to the Change Size command except that it also causes the VT to rescale the polygon points. For consistent implementation, the algorithm given below shall be used to calculate the new 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 signed integer algorithm:

```
if ( old_x < 0 ) then
        new_x = ((old_x * new_width) – (old_width / 2)) / old_width
else
        new_x = ((old_x * new_width) + (old_width / 2)) / old_width
endif

if ( old_y < 0 ) then
        new_y = ((old_y * new_height) – (old_height / 2)) / old_height
else
        new_y = ((old_y * new_height) + (old_height / 2)) / old_height
endif
```

Transmission repetition rate:	On request
Data length:	8
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 Polygon object to scale	
Bytes	4, 5			New width attribute	
Bytes	6, 7			New height attribute	
Byte	8			Reserved, transmit as FF_{16}	

F.55 Change Polygon Scale response

This message is available in VT Version 4 and later. The VT uses this message to respond to the Change Polygon Scale command.

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 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 = not used Bit 4 = 1 = Any other error	

F.56 Graphics Context command (transport protocol)

This message is available in VT Version 4 and later. This command is used to manipulate a graphics Context object.

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 those 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 is displayed in the viewport as a single pixel. The exact zooming algorithm is VT proprietary. A closest colour match may 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.

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 = 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	

See Table F.1.

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 Line object attribute. All drawing commands that follow use the new attribute value. For line suppression, set the object ID to NULL Object ID. The graphics cursor is not moved.	Bytes 5–6 = Object ID of a Line Attributes Object or NULL Object ID 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 Object ID. The graphics cursor is not moved.	Bytes 5–6 = Object ID of a Fill Attributes Object or NULL Object ID for no further filling	0 to 65534, 65535

Table F.1 (*continued*)

Sub-Command ID	Description	Parameters	Parameter range
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 Object ID. The graphics cursor is not moved.	Bytes 5–6 = Object ID of a Font Attributes Object or NULL Object ID for no Font Attributes	0 to 65534, 65535
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 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 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 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 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 of the bounding rectangle.	Bytes 5–6 = Width	0 to 65535
		Bytes 7–8 = Height	0 to 65535

Table F.1 (*continued*)

Sub-Command ID	Description	Parameters	Parameter range
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 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} NOTE All positions and offsets are signed integers unless otherwise noted.	–32768 to +32767
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 4.6.16.6).	
14	**Pan Viewport** This command modifies the viewport X and Y attributes and forces a redraw of the object, allowing “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.	Bytes 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
		Bytes 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. NOTE The size of the object (i.e. the memory used) cannot be changed. The graphics cursor is not moved.	Bytes 5–6 = New width	0 to 65535
		Bytes 7–8 = New height	0 to 65535

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 may 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 the colours allowed by this Graphics Context Object (see 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 the colours allowed by this Picture Graphic Object (see Table B.41) shall be treated as transparent.	Bytes 5–6 = Object ID of Picture Graphic object to copy to	0 to 65534
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 the colours allowed by this Picture Graphic Object (see 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

This message is available in VT Version 4 and later. The VT uses this message to respond to the Graphics Context command (transport protocol).

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 command id Bit 2 = 1 = Invalid parameter Bit 3 = 1 = Command will produce invalid results Bit 4 = 1 = Any other error	
Bytes	6–8			Reserved, transmit as FF_{16}	

F.58 Get Attribute Value message

This message is available in VT Version 4 and later. This command is used by a Working Set to query the VT for the current state of objects within the VT.

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, transmit as FF_{16}	

F.59 Get Attribute Value response

This message is available in VT Version 4 and later. The VT uses this message to respond to a Get Attribute Value message.

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 (0 = no error)	
		Bit 0 = 1 = Invalid object ID	
		Bit 1 = 1 = Invalid Attribute ID	
		Bits 2, 3 = Reserved, sent as zero	
		Bit 4 = 1 = Any other error	
Byte	8	Reserved, sent as FF_{16}	

F.60 Select Colour Map command

This message is available in VT Version 4 and later. The Select Colour Map command is used to select the active Colour Map. This command can take a long time to execute. The command applies to any presentation from the originating Working Set, which includes objects that may be shown on other Working Set screens (e.g. Auxiliary Control objects as may be presented on VT proprietary and other Working Set masks using the Auxiliary Control Designator Type 2 Object Pointer).

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
Bytes	2, 3			Object ID of the Colour Map object, or $FFFF_{16}$ to restore the default colour table (see A.3)	
Bytes	4–8			Reserved, transmit as FF_{16}	

If the selected Colour Map object is modified, the changes shall take effect immediately.

F.61 Select Colour Map response

This message is available in VT Version 4 and later.

Transmission repetition rate:	In response to Select Colour Map 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
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	
				Bit 2 = 1 = Any other error	
Bytes	5–8			Reserved, transmit as FF_{16}	

A VT can take a long time to process this command. The response message shall be delayed until this activity is completed. The VT Status message shall reflect the current state of the VT (is busy executing a command).

F.62 Identify VT message

This message is available in VT Version 4 and later. The Identify VT message may be sent by either Working Sets or VTs. 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. This message is intended to be sent Destination-Global; however, it may be sent destination-specific.

The presentation of the VT Number is considered VT proprietary, and the VT designer may choose to present other information indicating the purpose of the VT Number (see 4.6.21).

The VT Number shall be in the range of 1 to 32, corresponding to Function Instances 0 to 31. VTs may then be referenced as VT Number 1, VT Number 2, etc.

VT Number = VT Function Instance + 1

NOTE The offset of 1 is in support of operators that might not be familiar with a zero-based numbering system.

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, transmit as FF_{16}	

F.63 Identify VT response

This message is available in VT Version 4 and later. The VT uses this message to respond to the Identify VT message if it was received destination-specific.

Transmission repetition rate:	In response to Identify VT message
Data length:	8 bytes
Parameter group number:	VT to ECU, destination-specific

Byte	1	VT function = 187_{10}			
	Bits	7–4	1011	Command	Command
	Bits	3–0	1011	Parameter	Identify VT
Bytes	2–8			Reserved, transmit as FF_{16}	

Annex G
(normative)

Status Messages

G.1 General

The status messages allow the Working Set to determine the health of the VT and to monitor the progress of tasks in the VT. They also allow the VT to monitor the health of Working Sets. These messages are not part of object pools and are not allowed in macros.

G.2 VT Status message

The VT Status message shall be sent to the Global Address to declare the active Working Set Master which is both displayed AND has the input focus (it "owns" the VT).

For VT Version 4 and later, a Working Set Master shall listen to this message in conjunction with the VT On User-Layout Hide/Show message (see H.20) to determine if it is displayed by the VT.

Data traffic with the VT shall not proceed after initialization until the VT Status message is sent from the VT (see 4.6.6 for more information on connection management). The VT Status message is sent on change of any of Bytes 2 to 6, or Byte 7, bit 6, and once per second, up to a maximum of five per second.

Transmission repetition rate:	On change of any of Bytes 2 to 6, or Byte 7, bit 6, and once per second up to 5 messages per second
Data length:	8 bytes
Parameter group number:	VT to Global Address

Byte	1	VT function = 254_{10}		
	Bits 7–4	1111	Command	Status
	Bits 3–0	1110	Parameter	VT Status message
Byte	2		Source Address of active Working Set Master ("owns" VT)	
Bytes	3, 4		Object ID of the visible Data/Alarm Mask of the active Working Set	
Bytes	5, 6		Object ID of the visible Soft Key Mask of the active Working Set	
Byte	7	VT busy codes	VT busy codes Bit 0 = 1 = VT is busy updating visible mask Bit 1 = 1 = VT is busy saving data to non-volatile memory Bit 2 = 1 = VT is busy executing a command Bit 3 = 1 = VT is busy executing a Macro Bit 4 = 1 = VT is busy parsing an object pool[10] Bit 5 = Reserved, sent as zero Bit 6 = 1 = Auxiliary controls learn mode active[10] Bit 7 = 1 = VT is out of memory	
Byte	8	VT Function code	VT function code of current command being executed (valid only if command or Macro busy code is set)	

[10] These bits exist in VT Version 3 and later.

G.3 Working Set Maintenance message

Each Working Set Master shall send a Working Set Maintenance message cyclically each second to the VT (see 4.6.6 for more information on connection management).

As there are no means for a Working Set member to report its own version number, all members of the Working Set shall comply with the same version number as reported by the master. For example, a Working Set Master could report a Version Number that matches the lowest compliant device in the Working Set. The master and members may use proprietary messages to determine their compliance.

Transmission repetition rate:	Once per second
Data length:	8 bytes
Parameter group number:	ECU to VT, destination-specific

Byte	1	VT function = 255_{10}		
	Bits	7–4 1111	Command	Status
	Bits	3–0 1111	Parameter	Working Set Maintenance message
Byte	2	BitMask	Version 3 and later: Bit 0 = 1 = Initiating Working Set maintenance (once at startup) Bits 1–7 = Reserved, sent as zero Version 2: Reserved, sent as FF_{16}	
Byte	3	Version Number	The ISO 11783-6 version that this Working Set meets 0–2 3 4 5 – FE_{16} FF_{16}	 Reserved Compliant with VT Version 3 Compliant with VT Version 4 Reserved Compliant with VT Version 2 and prior
Bytes	4–8		Reserved, transmit as FF_{16}	

NOTE The Version Number reported in this message determines how the VT responds to commands and messages. See 4.6.2.

The Version Number parameter reported by the Working Set Master shall reflect the version of this part of ISO 11783 in conformance with which the Working Set (master and members) is designed. It shall not change at runtime due to adaptations to different VTs. For example, a Version 4 Working Set Master would still report Version 4 in this parameter, even if the Working Set falls back on Version 3 behaviour to upload an object pool to a Version 3 VT. The VT may choose to report this or provide it for diagnostics, but shall not reject communication or the object pool based on the reported Version Number.

Annex H
(normative)

Activation messages

H.1 General

Unsolicited messages are sent from the VT to the Working Set Master using the PGNs given in Annex C. The Working Set Master may send a response message (all the responses given in this annex are optional, unless specifically stated otherwise).

H.2 Soft Key Activation message

The Soft Key Activation message allows the VT to transmit operator selection of a Soft Key. If a key is held and the repetition rate exceeds 300 ms, then the ECU should process as if the key was released (see 4.6.15). If the VT has a means to abort the key press, it shall send the Key press aborted activation code instead of the Key press released activation code (e.g. press button on touch screen, slide finger off side of button, abort is sent).

Transmission repetition rate:	On key press/release and every 200 ms when key is held.
Data length:	8 bytes
Parameter group number:	VT to ECU, destination-specific

Byte	1	VT function = 0_{10}			
	Bits	7–4	0000	Command	Control Element Function
	Bits	3–0	0000	Parameter	Soft Key
Byte	2			Key activation code 0 = Key has been released (state change) 1 = Key has been pressed (state change) 2 = Key is still pressed 3 = Key press aborted[11]	
Bytes	3, 4	Object ID		Object ID of Key Object	
Bytes	5, 6	Parent object ID		Object ID of visible Data Mask, Alarm Mask, or in the case where the Soft Key is in a visible Key Group, the Object ID of the Key Group Object	
Byte	7	Key number		Soft key code: 0 = alarm ACK 1–255 = Key code assigned by Working Set Master	
Byte	8			Reserved, transmit as FF_{16}	

[11] VT Version 4 and later.

H.3 Soft Key Activation response

Transmission repetition rate:	Optionally, in response to Soft Key Activation message
Data length:	8 bytes
Parameter group number:	ECU to VT, destination-specific

Byte	1	VT function = 0_{10}		
	Bits	7–4 0000	Command	Control Element Function
	Bits	3–0 0000	Parameter	Soft Key
Byte	2	Key activation Code	Key activation code 0 = Key has been released (state change) 1 = Key has been pressed (state change) 2 = Key is still held 3 = Key press aborted[12)]	
Bytes	3, 4	Object ID	Object ID of Key Object	
Bytes	5, 6	Parent object ID	Object ID of visible Data Mask, Alarm Mask, or in the case where the Soft Key is in a visible Key Group, the Object ID of the Key Group Object	
Byte	7	Key number	Soft key code	
Byte	8		Reserved, transmit as FF_{16}	

H.4 Button Activation message

The Button Activation message allows the VT to transmit operator selection of a Button Object to the Working Set Master. If a non-latchable button is held and the repetition rate exceeds 300 ms, then the ECU should process as if the button was released (see 4.6.15). If the VT has a means to abort the button press, it shall send the Button press aborted activation code instead of the Button press released activation code (e.g. press button on touch screen, slide finger off side of button, abort is sent).

Transmission repetition rate:	On button press/release and every 200 ms when button is held. Latchable buttons do not repeat.
Data length:	8 bytes
Parameter group number:	VT to ECU, destination-specific

Byte	1	VT function = 1_{10}		
	Bits	7–4 0000	Command	Control Element Function
	Bits	3–0 0001	Parameter	Button
Byte	2	Key activation Code	Key activation code 0 = Button has been unlatched or released (state change) 1 = Button has been “pressed” or latched (state change) 2 = Button is still held (latchable buttons do not repeat) 3 = Button press aborted[12)]	
Bytes	3, 4	Object ID	Object ID of Button Object	
Bytes	5, 6	Parent object ID	Object ID of parent Data Mask or in the case where the Button is in a visible Window Mask object, the Object ID of the Window Mask object	
Byte	7	Key Number	Button key code	
Byte	8		Reserved, transmit as FF_{16}	

[12)] VT Version 4 and later.

H.5 Button Activation response

Transmission repetition rate:	Optionally, in response to Button Activation message
Data length:	8 bytes
Parameter group number:	ECU to VT, destination-specific

Byte	1	VT function = 1_{10}		
	Bits	7–4 0000	Command	Control Element function
	Bits	3–0 0001	Parameter	Button
Byte	2	Key activation code	Key activation code 0 = Button has been unlatched or released (state change) 1 = Button has been “pressed” or latched (state change) 2 = Button is still held (latchable buttons do not repeat) 3 = Button press aborted[13]	
Bytes	3, 4	Object ID	Object ID of Button Object	
Bytes	5, 6	Parent object ID	Object ID of parent Data Mask or in the case where the Button is in a visible Window Mask object, the Object ID of the Window Mask object	
Byte	7	Key number	Button key code	
Byte	8		Reserved, transmit as FF_{16}	

H.6 Pointing Event message

The Pointing Event message is used to indicate that a position in the Data Mask area has been touched, clicked or dragged, in the case where the VT supports a touch screen or pointing device.

This message is not used when a button or input object has been touched or clicked on, in which case the Button Activation message or VT Select Input Object message is sent.

If held and the repetition rate exceeds 300 ms, then the ECU should process as if released.

If the VT has a means to support dragging, and if the first press is not on a button or input field and the drag operation subsequently crosses a button or input field, this message shall continue to be sent. It shall not activate the button or input field.

If the VT has a means to detect changing coordinates while held, then the X,Y position shall update to reflect the current coordinates (see D.9).

If the VT does not have a means to detect changing coordinates while held, but can detect individual press and release coordinates, then the X,Y position while held shall repeat the coordinates at the press position (see D.9).

If the VT can only detect the press coordinates, then the X,Y position in all states is the coordinates at the press position (see D.9).

Transmission repetition rate:	On press/release and every 200 ms when held
Data length:	8 bytes
Parameter group number:	VT to ECU, destination-specific

[13] VT Version 4 and later.

Byte	1	VT function = 2_{10}		
	Bits 7–4	0000	Command	Control Element Function
	Bits 3–0	0010	Parameter	Pointing Event
Bytes	2, 3	X Position	X Position in pixels relative to top left corner of Data Mask area	
Bytes	4, 5	Y Position	Y Position in pixels relative to top left corner of Data Mask area	
Byte	6[14]		Touch State 0 = Released 1 = Pressed 2 = Held 255 = Pressed (For backward compatibility only)	
Bytes	7, 8		Reserved, transmit as FF_{16}	

H.7 Pointing Event response

Transmission repetition rate:	Optionally, in response to Pointing Event message
Data length:	8 bytes
Parameter group number:	ECU to VT, destination-specific

Byte	1	VT function = 2_{10}		
	Bits	7–4 0000	Command	Control Element Function
	Bits	3–0 0010	Parameter	Pointing Event
Bytes	2, 3	X Position	X Position in pixels relative to top left corner of Data Mask area	
Bytes	4, 5	Y Position	Y Position in pixels relative to top left corner of Data Mask area	
Byte	6[14]		Touch State 0 = Released 1 = Pressed 2 = Held 255 = Pressed (For backward compatibility only)	
Bytes	7, 8		Reserved, transmit as FF_{16}	

H.8 VT Select Input Object message

This message is sent by the VT any time an input field, Button or Key Object is selected (gets focus), deselected (loses focus), opened for edit or closed after edit by the operator or an ESC command.

This message may not be sent if the input object that was activated had a complete transaction applied to it as an atomic operation (e.g. an Input Boolean where the activation simply toggles the value, or an Input Number where the VT has a dedicated increment by 1 and decrement by 1 capability).

NOTE This command originates in the VT as a result of operator interaction. This message is not sent if the Working Set requested the change in focus with the Select Input Object command (see F.6).

When a Button or Key Object is the target of selection, it shall not be reported as open for input. VT Version 3 and prior do not support selection of a Button Object or a Key Object.

Transmission repetition rate:	On selection of input object
Data length:	8 bytes
Parameter group number:	VT to ECU, destination-specific

[14] Byte 6 applies to VT Version 4 and later, other than the value 255.

Byte	1	VT function = 3_{10}			
	Bits	7–4	0000	Command	Control Element Function
	Bits	3–0	0011	Parameter	VT Select Input object
Bytes	2, 3			Object ID	
Byte	4			Selection 0 = object is deselected, 1 = object is selected (has focus)	
Byte	5			VT Version 3 and prior Reserved, transmit as FF_{16} VT Version 4 and later Bitmask Bit 0 = 1 = object is open for data input – Byte 4 shall be set to 1 Bits 1–7 = Reserved, transmit as 0	
Bytes	6–8			Reserved, transmit as FF_{16}	

H.9 VT Select Input Object response

The ECU uses this message to optionally respond to the VT Select Input Object message.

Transmission repetition rate:	Optionally, in response to VT Select Input Object message
Data length:	8 bytes
Parameter group number:	ECU to VT, destination-specific

Byte	1	VT function = 3_{10}			
	Bits	7–4	0000	Command	Control Element Function
	Bits	3–0	0011	Parameter	VT Select Input object
Bytes	2, 3			Object ID	
Byte	4			Selection 0 = object is deselected, 1 = object is selected (has focus)	
Bytes	5–8			Reserved, transmit as FF_{16}	

H.10 VT ESC message

This message is sent by the VT any time the operator presses the ESC means, and when the VT closes an open input field due to a Change Active Mask command.

Transmission repetition rate:	On ESC means press
Data length:	8 bytes
Parameter group number:	VT to ECU, destination-specific

Byte	1	VT function = 4_{10}			
	Bits	7–4	0000	Command	Control Element Function
	Bits	3–0	0100	Parameter	VT 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 selected (this bit is only used when the VT has a permanent ESC means) Bits 1–3 = Not used, sent as 0 Bit 4 = 1 = Any other error	
Bytes	5–8			Reserved, transmit as FF_{16}	

H.11 VT ESC response

The ECU uses this message to optionally respond to the VT ESC message.

Transmission repetition rate:	Optionally, in response to VT ESC message
Data length:	8 bytes
Parameter group number:	ECU to VT, destination-specific

Byte	1	VT function = 4_{10}			
	Bits	7–4	0000	Command	Control Element function
	Bits	3–0	0100	Parameter	VT ESC
Bytes	2, 3			Object ID where input was aborted if no error code	
Bytes	4–8			Reserved, transmit as FF_{16}	

H.12 VT Change Numeric Value message

The VT sends this message any time the operator enters a numeric value for an input object or variable, regardless of whether or not the value changed. This message is not sent if the input was aborted (in which case a VT ESC message will be sent instead). For input objects that have a numeric variable reference, the object ID of the numeric variable object is used in this message.

Transmission repetition rate:	On change value of numeric object
Data length:	8 bytes
Parameter group number:	VT to ECU, destination-specific

Byte	1	VT function = 5_{10}			
	Bits	7–4	0000	Command	Control Element Function
	Bits	3–0	0101	Parameter	VT Change Numeric Value message
Bytes	2, 3			Object ID	
Byte	4			Not used	
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). Unused bytes shall be filled with zero.	
				Input Boolean:	1 byte for TRUE/FALSE
				Input Number:	4 bytes for integer input
				Input List:	1 byte for list index
				Number variable:	4 bytes for integer value

H.13 VT Change Numeric Value response

The ECU uses this message to optionally respond to the VT Change Numeric Value message.

Transmission repetition rate:	Optionally, in response to VT Change Numeric Value message
Data length:	8 bytes
Parameter group number:	ECU to VT, destination-specific

Byte	1	VT function = 5_{10}			
	Bits	7–4	0000	Command	Control Element Function
	Bits	3–0	0101	Parameter	VT Change Numeric Value message
Bytes	2, 3			Object ID	
Byte	4			Not used	
Bytes	5–8			Value copied from VT Change Numeric Value message.	

H.14 VT Change Active Mask message

The VT sends this message if there are missing object references or errors when a mask is displayed, or prior to deletion of pool due to active mask error. The Change Active Mask response message, also sent by the VT, only acknowledges that the Change Active Mask command was received and processed. Since the actual display of the mask can occur later, especially if an Alarm Mask currently has the display, this message is used by the VT to report errors during the actual drawing of the mask.

Transmission repetition rate:	On error
Data length:	8 bytes
Parameter group number:	VT to ECU, destination-specific

Byte	1	VT function = 6_{10}			
	Bits	7–4	0000	Command	Control Element Function
	Bits	3–0	0110	Parameter	VT Change Active Mask
Bytes	2, 3			Active mask object ID	
Byte	4			Error codes (0 = no error)	
				Bit 0 = not used	
				Bit 1 = not used	
				Bit 2 = 1 = Missing objects	
				Bit 3 = 1 = mask or child object has errors	
				Bit 4 = 1 = Any other error	
				Bit 5 = 1 = Pool being deleted	
Bytes	5, 6			Object ID containing error	
Bytes	7, 8			Parent object ID of error object ID	

H.15 VT Change Active Mask response

The ECU uses this message to optionally respond to the VT Change Active Mask message.

Transmission repetition rate:	Optionally, in response to VT Change Active Mask message
Data length:	8 bytes
Parameter group number:	ECU to VT, destination-specific

Byte	1	VT function = 6_{10}			
	Bits	7–4	0000	Command	Control Element Function
	Bits	3–0	0110	Parameter	VT Change Active Mask
Bytes	2, 3			Active mask object ID	
Bytes	4–8			Reserved, transmit as FF_{16}	

H.16 VT Change Soft Key Mask message

The VT sends this message if there are missing object references or errors when a Soft Key Mask is displayed. The Change Soft Key Mask response message, also sent by the VT, only acknowledges that the Soft Key Mask was received and processed. Since the actual display of the mask can occur later, especially if an Alarm Mask currently has the display, this message is used by the VT to report errors during the actual drawing of the Soft Key Mask.

Transmission repetition rate:	On error
Data length:	8 bytes
Parameter group number:	VT to ECU, destination-specific
Allowed in a Macro:	No

Byte	1	VT function = 7_{10}			
	Bits	7–4	0000	Command	Control Element Function
	Bits	3–0	0111	Parameter	VT Change Soft Key Mask
Bytes	2, 3			Data or Alarm Mask Object ID	
Bytes	4, 5			New Soft Key Mask Object ID	