INTERNATIONAL STANDARD

ISO 11783-6

First edition
2004-06-15

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 11783-6:2004(E)

Contents

Page

Foreword

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

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.

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

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 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 9: Tractor ECU*

Part 1, *General standard for mobile data communication*, Part 8, *Power train messages*, Part 10, *Task controller and management information system data interchange*, Part 11, *Data dictionary*, Part 12, *Diagnostics* and Part 13, *File server* are under preparation.

Introduction

Parts 1 to 11 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.

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

This part of ISO 11783 specifies a serial data network for control and communications on forestry or agricultural tractors, mounted, semi-mounted, towed or self propelled implements. Its purpose is to standardize the method and format of transfer of data between sensor, actuators, control elements, information storage and display units whether mounted or part of the tractor, or any implements. This part of ISO 11783 describes a universal virtual terminal that can be used by both tractors and implements.

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[1)], *Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 1: General standard for mobile data communication*

ISO 11783-3:1998, *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*

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 and abbreviated term apply.

3.1
auxiliary input device
autonomous electronic control unit (ECU) providing auxiliary controls for common use that may also be being physically located on the virtual terminal (VT)

1) Under preparation.

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.

AID attribute ID

4 Technical requirements

4.1 Overview

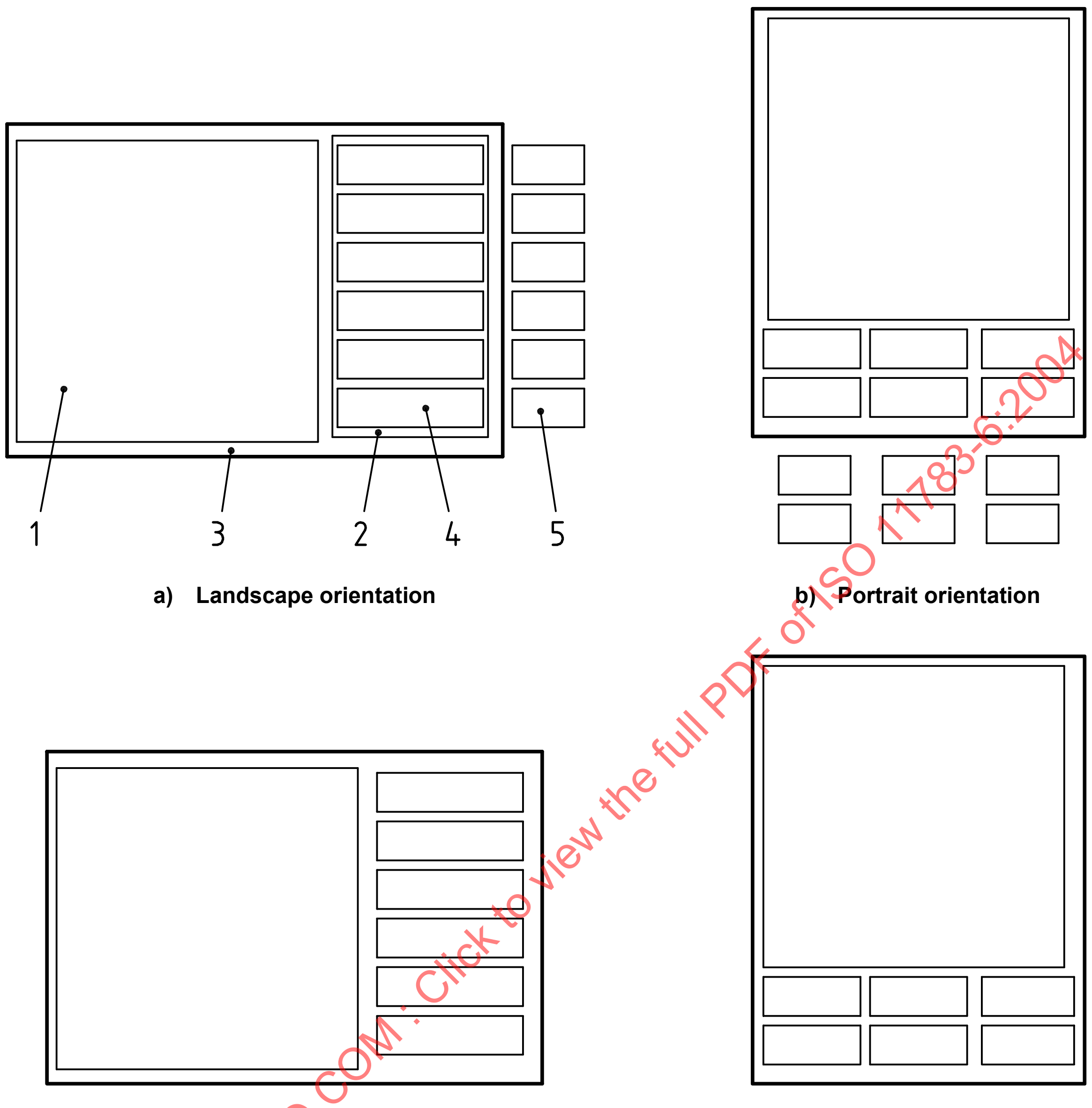
A virtual terminal (VT) is an electronic control unit (ECU), consisting of a graphical display and input functions, connected to an ISO 11783 network that provides the capability for an ECU, an implement or a group of implements to interact with an operator. The VT provides the capability to display information and to retrieve data from an operator. An ECU, 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 an ECU, an implement or a group of implements either represented by a single ECU or a group of ECUs acting as a working set. Working sets on the network can also acquire the use of input and control keys of the VT to allow the operator to send signals back to the working set.

This part of ISO 11783 describes a VT with the detail and clarity required for VTs built by different manufacturers to be interchangeable with any implement working set that uses its services. The interface protocol of this part of ISO 11783 also reduces the run-time 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 must be imposed in order to meet the goal of interchangeability between various manufacturers. Specifications regarding physical layout, components, processing power and the number of physical elements comprising a VT have been omitted in order to avoid restricting manufacturer's designs.

The VT shall have a pixel-addressable (graphical) display. Information from connected working sets is shown to the operator on the graphical display. This information is shown in display areas that are defined by data masks, alarm masks and soft key masks. The data for these masks is contained in object definitions that are loaded into a VT either from a working set via the ISO 11783 CAN bus, from a data card, or by some other means. When the information defined by a mask is required on the display, the mask can be made visible by a single Change Active Mask message 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.



a) **Landscape orientation**

b) **Portrait orientation**

c) **Touch screen — Landscape orientation**

d) **Touch screen — Portrait orientation**

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.

a) **Soft**

is a means, most likely keys on the VT, using software-changeable designators (labels). “Soft keys” have their identity changed depending on which soft key mask is visible. The VT shall make the association between a soft key and its designator clearly evident to the operator. There is no requirement on the number of physical soft keys. The number of soft key designators supported by the VT shall be between 6 and 64 inclusive per soft key mask. The VT shall provide a means for the operator to navigate and select all defined soft key designators. For example, if there are four physical keys but the VT design supports 16 soft key designators, some type of scrolling or paging would be required to allow the operator to navigate to, and select from, any of the 16 soft key designators using the four physical keys.

b) **Navigation**

is a means of selecting an input field within the active data or alarm mask. “Navigation keys” do not send key activation information to the working set and are proprietary to the VT.

c) **Editing**

is a means of entering/editing information in an input field. “Editing keys” 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. A means shall also be provided for ESC from or ENTER information into a data field. The ENTER means shall be provided to indicate to ECUs the completion of data entry and the ESC means shall be provided to indicate that the data entry was aborted. The VT shall send a response message to an ECU for either an operator-activated ESC means or an ESC command received from an ECU.

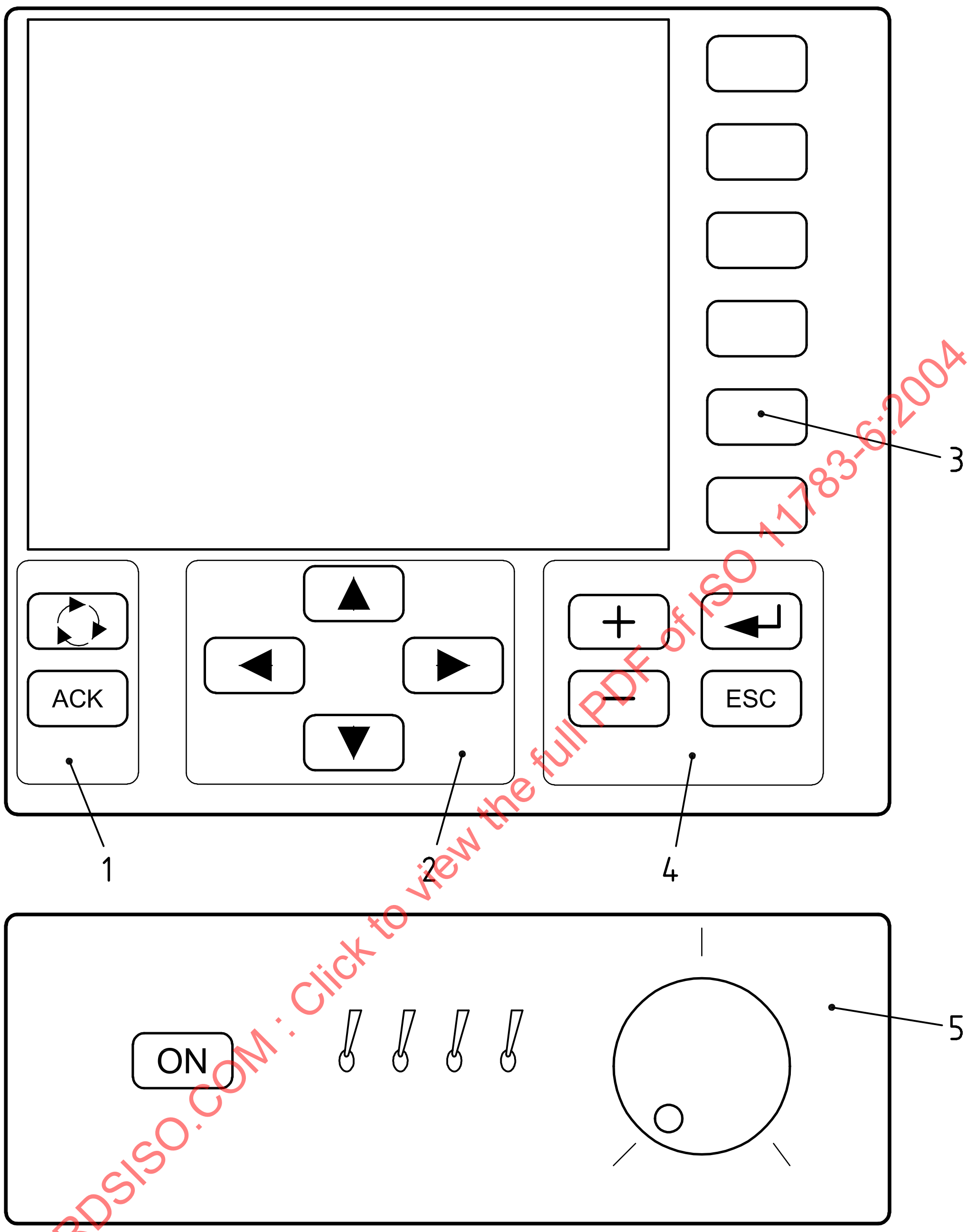
d) **Control**

is a means of selecting between working sets whenever a data mask is visible and for acknowledging alarms. Both means are required. Since more than one working set can use the services of the VT, the VT shall provide a means for the operator of selecting between connected working sets. The working set selection means shall be indicated by three circular arrows or a similar graphic. Only the ACK means sends key activation information to the working set.

e) **Auxiliary input**

is a means for assigning auxiliary inputs to auxiliary functions. See Annex J.

See Figure 2.



Key

1	control	4	editing
2	navigation	5	auxiliary input
3	soft		

Figure 2 — Operator input and control means

4.3 Acoustic alarm

The VT shall provide an acoustic alarm. The alarm may be a simple on/off type buzzer or an acoustic device 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) is located at the top left-hand corner of the data mask area.

4.5 Display areas

4.5.1 Data mask area

The VT shall reserve an area of the display for displaying data 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 device, only the useable data mask area. Higher resolution mask areas are permitted, but the square aspect ratio shall be strictly enforced. Examples of data mask areas that would meet this requirement are

— 200 × 200,

— 240 × 240,

— 320 × 320, 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. If a soft key is continually pressed after another mask has been activated, the VT should not send a Key Activation message for the new mask. It should send the Release Key message for the pushed soft key for the previous mask.

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 clauses. 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 is 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 is associated with all members of that working set. This allows object information from one ECU or all the ECUs 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 of all working set members. 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 ECUs which can provide data to the fields in the working set's masks. Appropriate messages for defining a working set are given in ISO 11783-7.

Once members of the working set have been identified to the VT, any member of the working set has the ability to provide data for output objects and to change attributes in the object pool during run-time. The VT shall be able to accept the change attribute type commands from any member of a working set.

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.

4.6.3 Language, formats and measurement units selection

The VT shall send the standard language, format and measurement units messages defined in ISO 11783-7. 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 — by updating string fields — and to the selected units — by changing offsets and scales. If the working set does not support the specified language, formats or units, it should use a proprietary method to select an appropriate setting.

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

4.6.4 Initialization

Upon power-up, 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.

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.

5) The VT shall complete the auxiliary input initialization defined in Annex J.

b) **Working set initialization**

1) The working set master shall complete the address claim procedure in accordance with ISO 11783-5.

2) The working set master shall wait until the VT begins transmission of the VT status message.

3) The working set master shall identify itself and its members to the VT using messages given in ISO 11783-7.

4) The working set master shall begin transmitting the working set maintenance message.

5) The working set master shall request the language and format messages from the VT (see ISO 11783-7).

6) 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.

7) The working set master may query the VT to determine if its object pool already exists in non-volatile memory.

8) 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.

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 ownership of the VT display. The VT shall provide some means to allow the operator to select 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 another working set is active and it cannot force its working set object to be active. Whenever a working set becomes active, the VT shall

a) display the new working set's current active mask and associated soft key mask, and

b) send the VT status message to the global address (255) to inform working sets of the change.

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 not the active working set. This allows the appropriate mask to be displayed when the working set becomes active. When a working set is deactivated, its active mask is hidden, 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. ECUs use the message to ensure the VT is present and to determine the current status of the VT. If an ECU does not receive this message for at least 3 s it assumes a possible uncontrolled shutdown of the VT. When this happens the working set shall enter a safe state. The safe state is defined as the state in which all functions dependant on the VT operator interface are put into a known state that will not put the operator or machine at risk. The working set shall not leave the safe state until power to the working set is cycled.

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 at least 3 s it assumes a possible uncontrolled shutdown of the working set master. When this happens the VT shall delete the working set's object pool to free the memory for other uses. If the working set has control of the VT, the display is cleared and the VT automatically gives control to another connected working set and sends 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 the VT receives a Working Set Maintenance message from the missing working set, it should NACK the message (ISO 11783-3). The NACK message is sent to the working set's source address. When a working set has been disconnected and reconnected to the VT, the working set may attempt to reload its object pool.

4.6.7 Updating the operator interface

4.6.7.1 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 (AID). The Change Attribute message allows any attribute with an AID to be changed. 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.

4.6.7.2 Changing, adding and deleting objects

Objects can be completely redefined at run-time 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 may cause the VT to run out of memory. See Annex C.

The entire object pool can be deleted by the working set using a Delete Object Pool message.

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) to hide and show objects.

Containers create hierarchy within a mask.

Figure 3 shows an example of container re-use. 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.

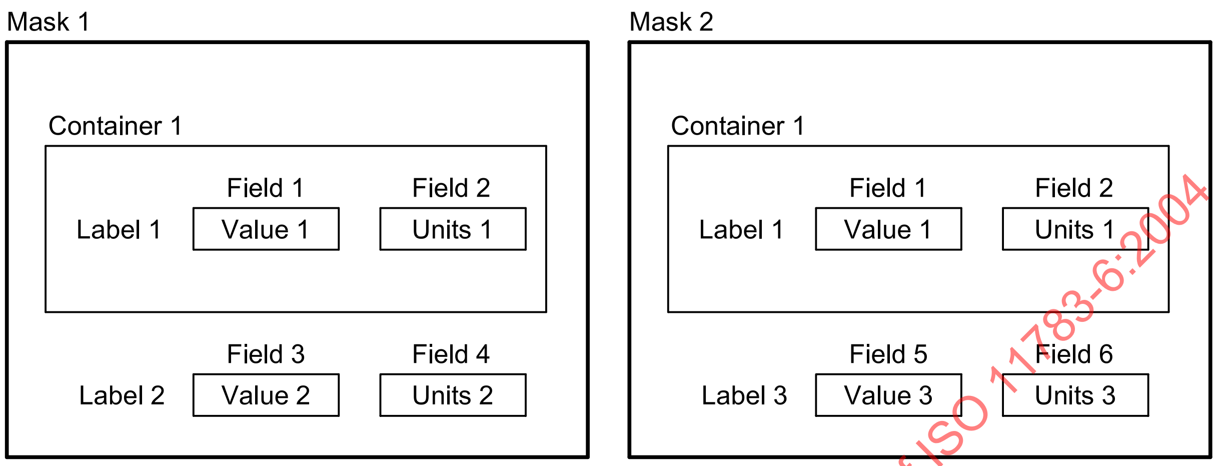
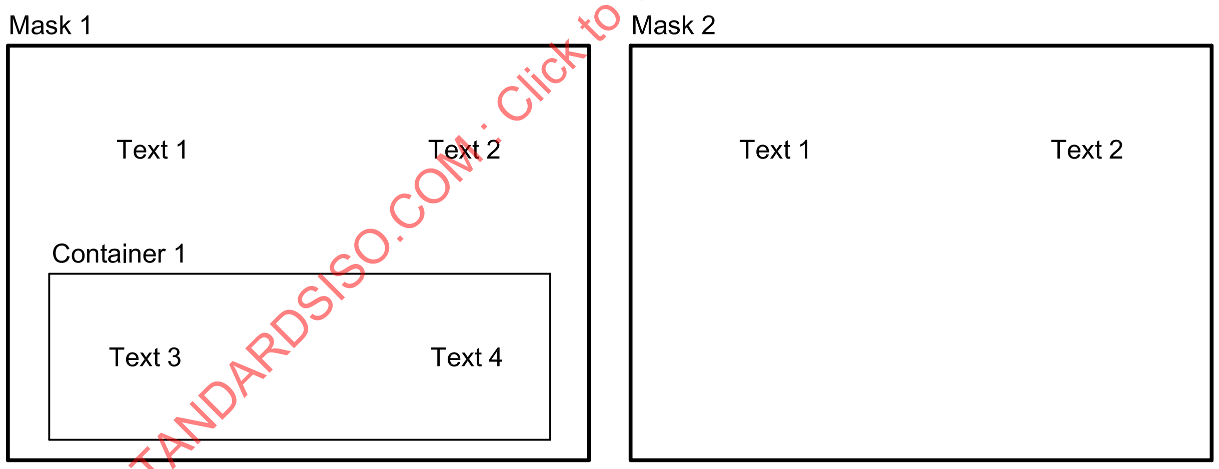


Figure 3 — Container re-use

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



a) **Particular feature available**

b) **Feature not available: container hidden**

Figure 4 — Container used to hide objects — Example

4.6.8.2 Attribute objects

There are four types of attribute object: font, line, fill and 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 might 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 Value message.

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 in the macro.

Circular references are not permitted since they create infinite macro loops inside the VT and could render the VT inoperable for all working sets.

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

4.6.8.5 Object pointer

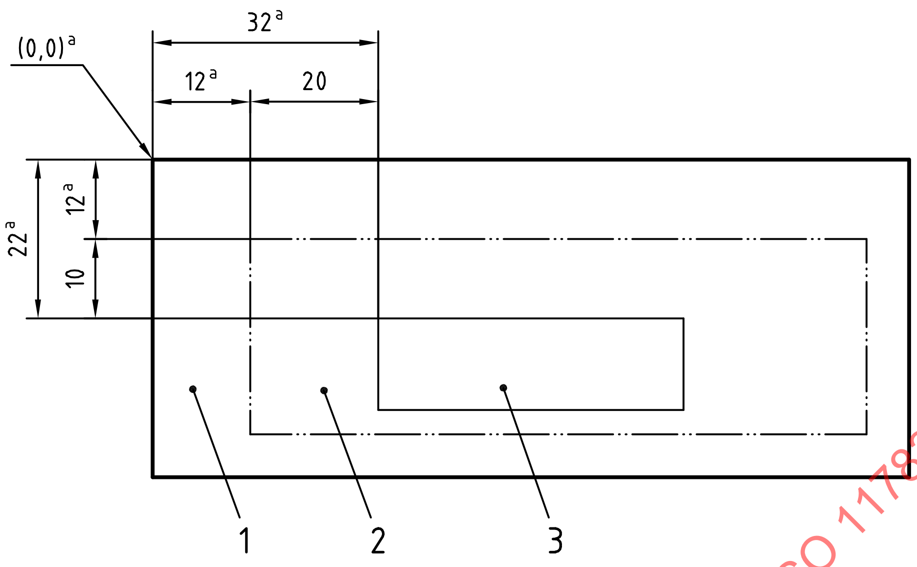
The object pointer object allows run-time modification of included objects. By changing the value of the pointer object, a different object can be drawn at the same location. The type of object that the object pointer is allowed to reference is limited and depends on the parent object.

When an object pointer is modified at run-time, 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 or alarm mask which contains the object pointer is activated. At activation of a data or alarm mask containing the invalid object reference the VT sends the F.35 “Change Active Mask Response” or H.14 “VT Change Active Mask” messages 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 5 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.

Figure 5 — Relative and absolute location of objects

4.6.10 Overlaid objects

A mask may be built such that two objects will occupy the same or overlapping space on the display. It needs to be recognized that this will 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. Each mask and container object has 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. The VT first deletes the object and all child objects by filling the object's area with the background colour of the parent mask. The VT shall then redraw the object along with all child objects. The VT shall then refresh any other object that may have been damaged or destroyed in the deletion process. Figure 6 shows an example. If the working set hides Object 1, the result is shown in b). The VT shall then refresh Object 2.

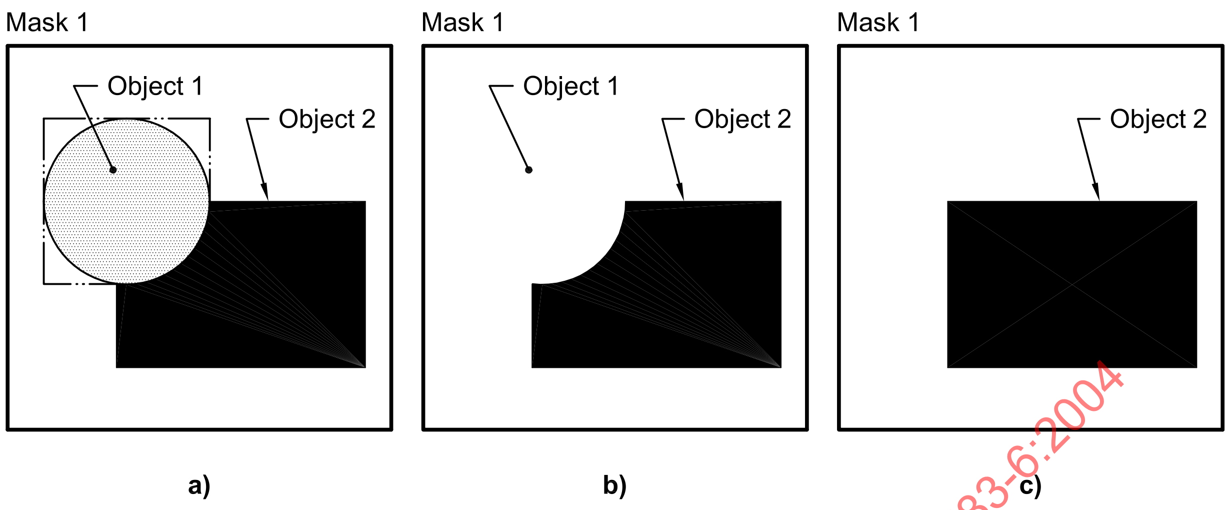


Figure 6 — Object changed or hidden — Display update

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.

Operator input disturbed by the activation of an alarm mask should 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 message on the working set object. Only one mask can be active per working set.

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

c) Based on priorities, at some point, the VT displays the alarm mask and the soft key mask associated with the alarm mask. See Table 1.

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 message on the working set object.

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

h) The VT displays a new mask. See Table 1.

Table 1 — VT behaviour on mask transition

Working set's active mask attribute From/To	Is requester the Current active working set?	VT behaviour
Data to data	Yes	Hide current data mask, show new data mask.
Data to data	No	No events triggered.
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 events triggered.

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.

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 before transmission of its object pool to the VT. 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 fit 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

The VT input objects control the input of strings, Booleans, lists and numbers from an operator and transfer the data from the VT to the working set master. It is the responsibility of the VT to indicate to the operator the presence of input fields and to provide mechanisms for

— moving to any input field (navigation means),

— initiating input (editing means),

— manipulating the input data (editing means), and

— completing or terminating the input (editing means).

The steps that occur during input are the following.

a) Whenever the VT displays a data mask that contains one or more enabled, visible, input objects, operator input is possible. Note that when determining the visibility of an input object, the hierarchy of parent objects shall be considered. For example, if the input object is part of a container and the container is hidden, the input object is not accessible. In addition, input objects may be visible but disabled. The VT shall provide some indication to the operator when an input field is disabled.

   EXAMPLE The input object could be set to a distinctive colour.

b) The VT displays the data mask and provides a proprietary means for the operator to navigate through enabled and visible input objects and to select the input object of choice. Tab order for navigating to the various input fields is defined by the definition order of the input fields in the parent object.

c) As the operator navigates to each input field the VT sends a VT Select Input Object message to the working set master.

d) The VT provides proprietary means for the operator to edit and change the value of the input object. When the operator has finished changing a value and has saved it with the ENTER means, the VT sends a Change Numeric Value response or Input String Value message to the working set.

e) If the input is aborted, either by the operator using the ESC means or by the working set sending an ESC message, the value of the input field is restored to its original value and the VT sends an ESC response message to the working set.

f) If a field is being modified and then the operator navigates to a new input object without completing or terminating the current input object, it is treated as an ESCape and the procedure in step e) is used.

g) The working set may force selection of an input field in the active (visible) data mask by sending a Select Input Object message. However, if the operator has already started entering data into an object, the VT ignores the command but responds with a Select Input Object response message with the object ID from the Select Input Object command and with an error indicating that another input field is currently being modified. If this occurs, the working set can override by first sending an ESC command followed by the Select Input Object message.

h) The working set may disable an input field with an Enable/Disable Object message. If the input field is selected but no data entry has started, the VT shall deselect the field. If the input field is selected and data entry has started, the VT ignores the command but responds with a Enable/Disable Object response message with the object ID from the Enable/Disable Object command and with an error indicating that the operator is entering data. If this occurs, the working set can override by first sending an ESC command followed by the Enable/Disable Object message.

In addition to the above, the VT shall correctly implement the input object events described with the objects.

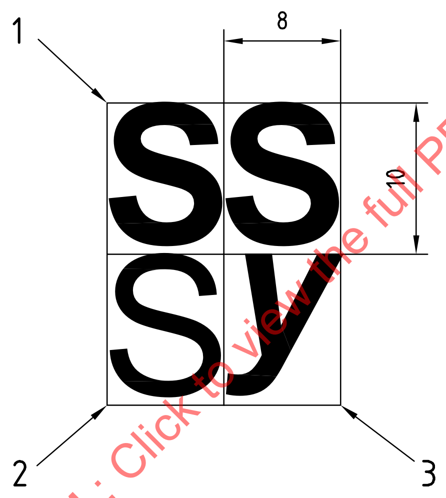
4.6.15 Soft key and button activation

Whenever a soft key or button object or ACK key is pressed, released, or latched, the VT sends a Soft Key Activation or Button Activation message to the owner working set. 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.

4.6.16 Font rendering

The VT uses non-proportional block fonts only. 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 cannot exceed the size of the box defined by the font size, regardless of style. For example, an 8 × 10 font set to bold and italics shall still fit inside the 8 × 10 pixel area. Space should 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 7.

Dimensions in pixels



Key

1 bold

2 normal (upright)

3 bold italic

Figure 7 — 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.

4.6.17 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.

Pattern fills of output shape objects shall be done as follows.

a) The pattern shall be a picture graphic object whose width is 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-hand corner of the pattern buffer is anchored to the upper left-hand corner of the VT's physical data mask area and repeats across and also down. This rule ensures that the pattern matches between objects in the data mask area and that the pattern fill looks the same on all VT designs.

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

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 a macro associated with it, the macro is executed by the VT. 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 A soft key press event could cause an appropriate data mask to be made active.

4.6.19 Touch screens and pointing devices

The VT design can optionally support a touch screen or pointing device 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 object has been touched or clicked on.

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	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	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	2_{10}	Top level object that contains other objects. Describes an alarm display.
Container	3_{10}	Used to group objects.
Key objects		
Soft key mask	4_{10}	Top level object that contains key objects.
Key	5_{10}	Used to describe a soft key.
Button	6_{10}	Used to describe a button control.
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.

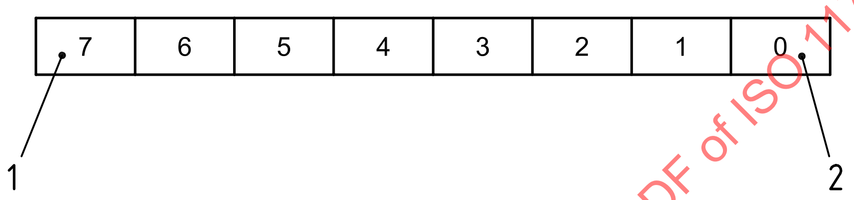
Table A.1 (*continued*)

Object	Type ID	Description
Output shape objects		
Line	13_{10}	Used to output a line.
Rectangle	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	17_{10}	Used to output a meter.
Linear bar graph	18_{10}	Used to output a linear bar graph.
Arched bar graph	19_{10}	Used to output an arched bar graph.
Picture graphic object		
Picture graphic	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 object	23_{10}	Used to group font based attributes. Can only be referenced by other objects.
Line attributes object	24_{10}	Used to group line based attributes. Can only be referenced by other objects.
Fill attributes object	25_{10}	Used to group fill based attributes. Can only be referenced by other objects.
Input attributes object	26_{10}	Used to specify a list of valid characters. Can only be referenced by input field objects.
Pointer object		
Object pointer	27_{10}	Used to reference another object.
Macro object		
Macro	28_{10}	Special object that contains a list of commands that can be executed in response to an event. Macros can only be referenced by other objects.
Auxiliary control		
Auxiliary function	29_{10}	The auxiliary function object defines the designator and function type for an auxiliary function. The object is used strictly in the auxiliary control screen which is proprietary to the VT.
Auxiliary input	30_{10}	The auxiliary input object defines the designator, key number, and function type for an auxiliary input. The object is used strictly in the auxiliary control screen which is proprietary to the VT.

A.1.2 Nomenclature

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

Boolean	Logical TRUE (1) or FALSE (0). Size is 1 byte.
Integer	Signed or unsigned integer numeric value. Possible sizes are 1, 2 or 4 bytes.
Float	IEEE 754-1985 standard 32-bit floating point numeric value. Size is 4 bytes.
Char	A single text character. Size is 1 byte.
String	Zero or more text characters. Size is variable.
Bitmask	A set of logical bit values. Size is 1 byte. Bitmasks always have Bit 0 defined as the least significant bit. See Figure A.1.



Key

1 most significant bit

2 least significant bit

Figure A.1 — Bit positions in a bitmask

“may contain”	Indicates object types that can be listed as child objects.
“may reference”	Indicates object types that are referenced in the object’s attributes.

A.1.3 Object relationships

The relationships between objects are shown in Figures A.2, A.3, A.4 and A.5.

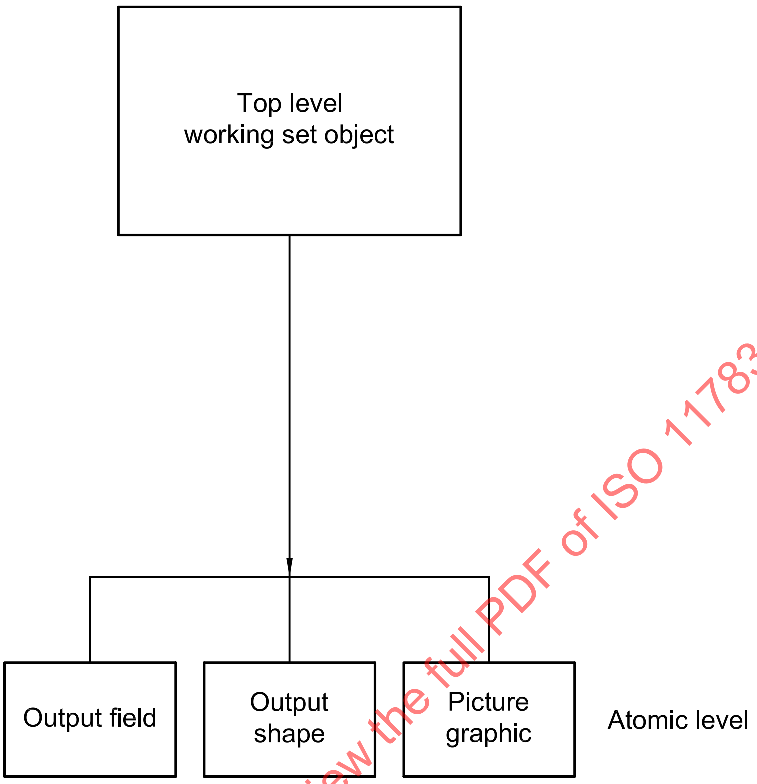


Figure A.2 — Working set object relationships

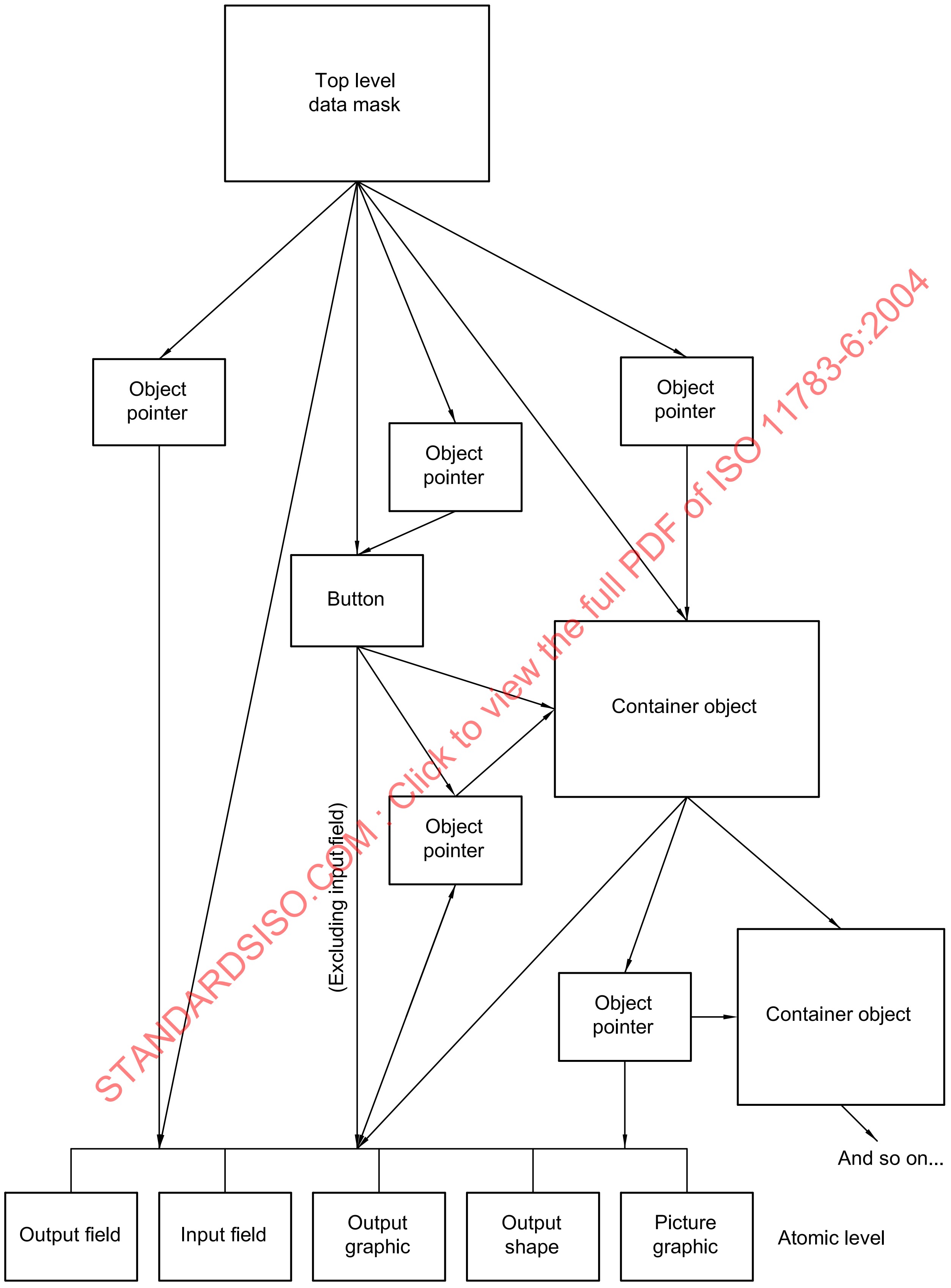


Figure A.3 — Data mask object relationships

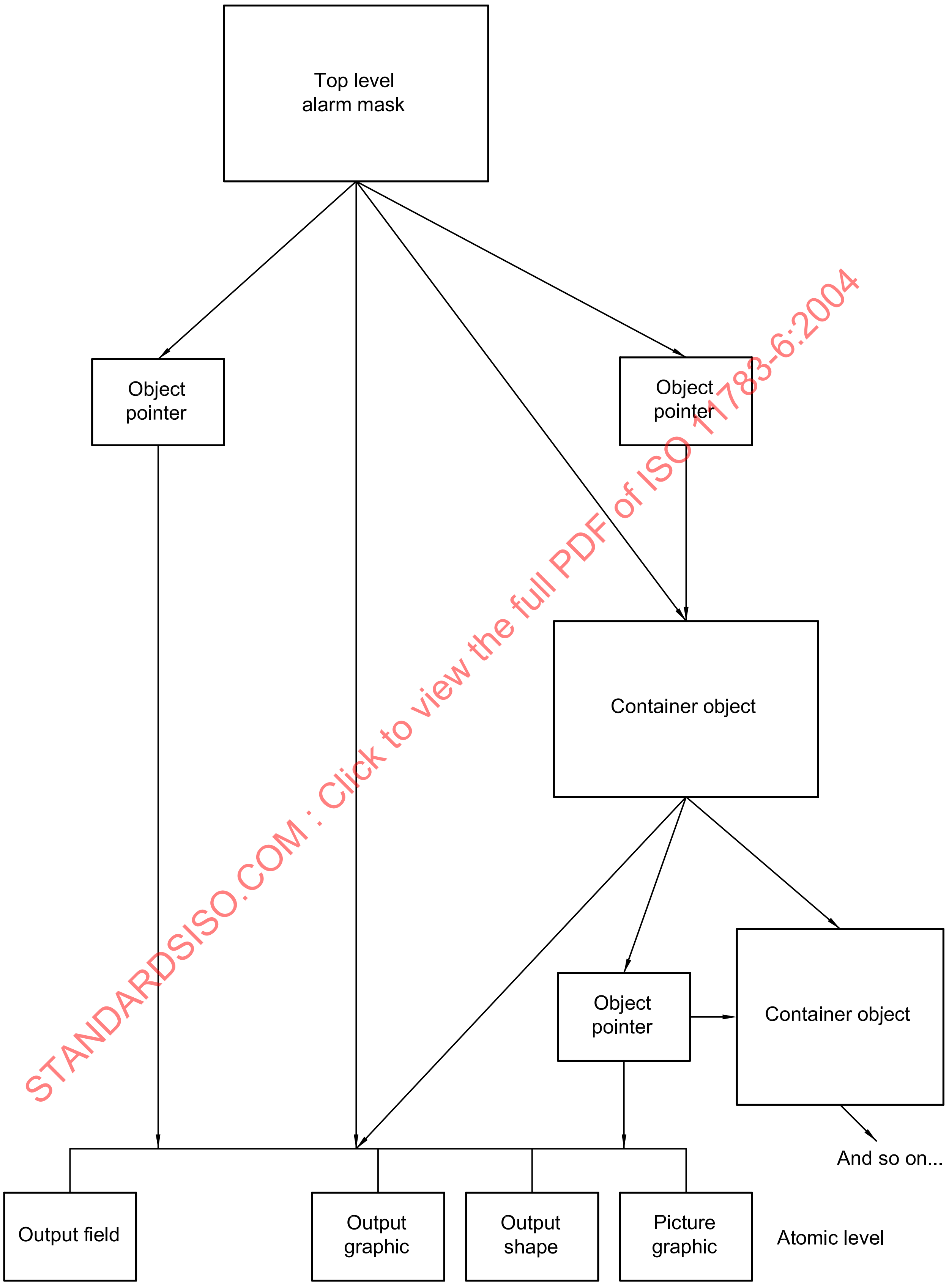


Figure A.4 — Alarm mask object relationships

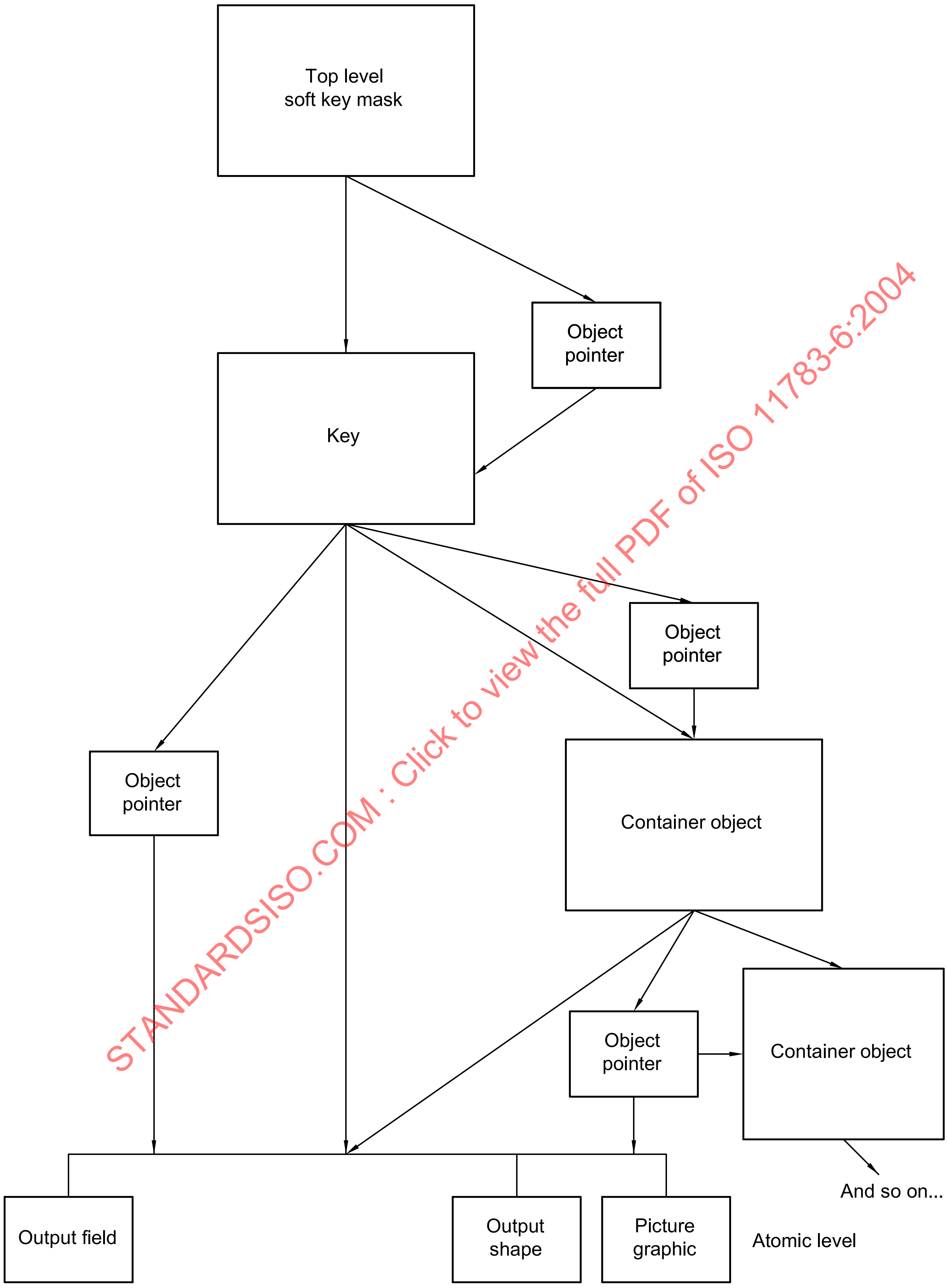


Figure A.5 — Soft key mask object relationships

A.2 Event types

Table A.2 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.2 — Event summary

Event name	Event ID	Event occurs when:
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}	Change Soft Key Mask command.
On Change Attribute	9_{10}	Change Attribute Command.
On Change Background Colour	10_{10}	Change Background Colour command.
On Change Font Attributes	11_{10}	Change Font Attributes command.
On Change Line Attributes	12_{10}	Change Line Attributes command.
On Change Fill Attributes	13_{10}	Change Fill Attributes command.
On Change Child Location	14_{10}	Change Child Location command.
On Change Size	15_{10}	Change Size Command.
On Change Value	16_{10}	Change Numeric Value or Change String Value Command.
On Change Priority	17_{10}	Change Priority command.
On Change End Point	18_{10}	Change End Point command.
On input field selection	19_{10}	The input field has received focus, operator has navigated onto the input field.
On input field deselection	20_{10}	The input field has lost focus, operator has navigated off of the input field.
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}	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.

A.3 VT standard colour palette

See Table A.3.

Table A.3 — 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[a, b]	64	33,66,00	128	99,00,CC	192	CC,FF,66
1 (White)	FF,FF,FF [a, b]	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.3 (*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 to extend 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 this palette table.

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

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

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

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

See Table A.3.

A.4 Command/parameter code summary

Table A.4 gives all defined messages with the corresponding function code.

Table A.4 — Command/parameter summary

Clause/ subclause	Message	Direction	Function (Decimal)	Function (Hex)	Allowed in macro
C.2.3	Object Pool Transfer (Transport Protocol)	ECU to VT	17_{10}	11_{16}	No
C.2.4	End of Object Pool	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	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	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	ECU to VT	199_{10}	$C7_{16}$	No
D.9	Get Hardware Response	VT to ECU	199_{10}	$C7_{16}$	No
E.2	Get Versions	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	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	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	ECU to VT	210_{10}	$D2_{16}$	No
E.9	Delete Version Response	VT to ECU	210_{10}	$D2_{16}$	No

Table A.4 (*continued*)

Clause/ subclause	Message	Direction	Function (Decimal)	Function (Hex)	Allowed in macro
F.2	Hide/Show Object	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	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	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	ECU to VT	146_{10}	92_{16}	No
F.9	ESC Response	VT to ECU	146_{10}	92_{16}	No
F.10	Control Audio Device	ECU to VT	163_{10}	$A3_{16}$	Yes
F.11	Control Audio Device Response	VT to ECU	163_{10}	$A3_{16}$	No
F.12	Set Audio Volume	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	ECU to VT	165_{10}	$A5_{16}$	Yes
F.15	Change Child Location Response	VT to ECU	165_{10}	$A5_{16}$	No
F.16	Change Child Position (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	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	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	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 (Transport Protocol)	ECU to VT	179_{10}	$B3_{16}$	Yes
F.25	Change String Value Response	VT to ECU	179_{10}	$B3_{16}$	No
F.26	Change End Point	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	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	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	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	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	ECU to VT	174_{10}	AE_{16}	Yes
F.37	Change Soft Key Mask Response	VT to ECU	174_{10}	AE_{16}	No

Table A.4 (*continued*)

Clause/ subclause	Message	Direction	Function (Decimal)	Function (Hex)	Allowed in macro
F.38	Change Attribute	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	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	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	ECU to VT	178_{10}	$B2_{16}$	No
F.45	Delete Object Pool Response	VT to ECU	178_{10}	$B2_{16}$	No
G.2	VT Status	VT to global address	254_{10}	FE_{16}	No
G.3	Working Set Maintenance	ECU to VT	255_{10}	FF_{16}	No
H.2	Soft Key Activation	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	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	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	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	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	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	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	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	Input String Value (Transport Protocol)	VT to ECU	8_{10}	08_{16}	No
H.19	Input String Value Response	VT to ECU	8_{10}	08_{16}	No
J.6.2	Auxiliary Assignment Command	VT to ECU	32_{10}	20_{16}	No
J.6.3	Auxiliary Assignment Response	ECU to VT	32_{10}	20_{16}	No
J.6.4	Auxiliary Input Status	ECU to global address	33_{10}	21_{16}	No

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.

a) **Relationships:**

— **may contain** output field, output shape, picture graphic;

— **may reference** data mask, alarm mask, macro;

— **may be contained by** none.

b) **Allowed commands:**

— Change Active Mask;

— Change Background Colour;

— Change Child Location.

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
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.	Change 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).	Change Child Location Response
On Change Child Position	Change Child Position command	Draw child object at current location in background colour to erase it. Refresh Data Mask (to redraw child object or objects).	Change Child Position Response

Table B.2 — Working set attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=0	3	Object Type = Working Set
Background colour		Integer	1	0-255	4	Background colour.
Selectable		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 devices that do not have any data masks should not be made selectable.)
Active mask		Integer	2	0-65534	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	1-255	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	0-255	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	0-255	10	The number of language codes to follow. Each two-letter code represents a language that the working set can support.
Repeat: {Object ID}		Integer	2	0-65534	11+object*6	Working set identifier (shall fit inside soft key designator) Object ID of a picture graphic, output shape or output field object.
{X Location}		Signed integer	2	– 32767 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	– 32767 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	0-255	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	0-255	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.

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 to fill the entire data mask area of the physical display. For this reason, no size attribute is required. See Tables B.3 and B.4.

a) **Relationships:**

— **may contain** output field, input field, output graphic, output shape, picture graphic, button, container (containing only objects on this list), object pointer (referencing only objects on this list);

— **may reference** soft key mask, macro;

— **may be contained by** none (top level).

b) **Allowed commands:**

— Change Background Colour;

— Change Child Location;

— Change Child Position;

— Change Soft Key Mask;

— Change Attribute.

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
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	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 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.
On Change Attribute	Change Attribute command	See change background colour and change soft key mask behaviour above.	Change Attribute Response. If change affects visible masks, VT Status.

Table B.4 — Data mask attributes and record format

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record byte**	**Description**
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=1	3	Object Type = Data mask
Background colour	1	Integer	1	0-255	4	Background colour.
Soft key mask	2	Integer	2	0-65534 or 65535	5-6	Object ID of a soft key mask associated with this data mask. Whenever this data mask is displayed, the associated soft key mask is also displayed. If the NULL object ID is used, there are no soft keys associated with this data mask and the soft key designators should be cleared.
Number of objects to follow		Integer	1	0-255	7	Number of objects to follow even if zero. Each of these objects is “contained” in this data mask. Each object consists of 6 bytes: two (2) for object ID and four (4) for location.
Number of macros to follow		Integer	1	0-255	8	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Object ID}		Integer	2	0-65534	9+object*6	Object ID of an object contained in this mask. List all objects before listing macros.
{X Location}		Signed integer	2	– 32767 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	– 32767 to + 32767	13+object* 6	Relative Y location of the top left corner of the object (relative to the top left corner of the data mask).
Repeat: {Event ID}		Integer	1	0-255	9+(No. objects *6)…	(List these after all objects have been listed) Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	10+(No. objects *6)…	Macro ID of the macro to execute.

B.3 Alarm mask object

For information on alarm mask behaviour, see Clause 4.6.11. See Tables B.5 and B.6.

a) **Relationships:**

— **may contain** output field, output graphic, output shape, picture graphic, container (containing only objects on this list), object pointer (referencing only objects on this list);

— **may reference** soft key mask, macro;

— **may be contained by** none.

b) **Allowed commands:**

- Change Background Colour;
- Change Child Location;
- Change Child Position;
- Change Priority;
- Change Soft Key Mask;
- Change Attribute.

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
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	Draw child object at current location in background colour to erase it. Refresh alarm Mask (to redraw child object or objects).	Change Child Position Response
On Change Priority	Change Priority command	If this is the current mask in this working set then reevaluate alarm priorities as follows a) If this alarm mask is visible and it is no longer the highest priority alarm then deactivate this working set and activate the WS with the highest priority alarm b) If this alarm mask is not visible and it becomes the highest priority alarm then deactivate the current working set and activate this WS.	Change Priority Response
On Change Soft Key Mask	Change Soft Key Mask command	If the alarm mask is visible, hide event on the current soft key mask and a show event on the new soft key mask.	Change Soft Key Mask Response. If this mask is visible, VT Status.
On Change Attribute	Change Attribute command	For behaviour see other change commands above.	Change Attribute Response. If change affects visible masks, VT Status.

Table B.6 — Alarm mask attributes and record format

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record byte**	**Description**
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=2	3	Object Type = Alarm Mask
Background colour	1	Integer	1	0-255	4	Background colour.
Soft key mask	2	Integer	2	0-65534 or 65535	5-6	Object ID of a soft key mask associated with this alarm mask. Whenever this alarm mask is displayed, the associated soft key mask is also displayed. If the NULL 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	0-2	7	Priority of this alarm as follows: 0 = High, operator is in danger or urgent machine malfunction 1 = Medium, normal alarm, machine is malfunctioning 2 = Low, information only
Acoustic signal	4	Integer	1	0-3	8	Acoustic signal. 0 = highest priority, 1 = medium priority, 2 = lowest priority, 3 = none (silent).
Number of objects to follow		Integer	1	1-255	9	Number of objects to follow even if zero. Each of these objects is “contained” in this alarm mask. Each object consists of 6 bytes: two (2) for object ID and four (4) for location.
Number of macros to follow		Integer	1	0-255	10	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Object ID}		Integer	2	0-65534	11+object*6	Object ID of an object contained in this mask. List all objects before listing macros.
{X Location}		Signed integer	2	– 32767 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	– 32767 to + 32767	15+object*6	Relative Y location of the top left corner of the object (relative to the top left corner of the alarm mask).
Repeat: {Event ID}		Integer	1	0-255	11+(No. objects *6)…	(List these after all objects have been listed.) Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	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 run-time under working set control. The container object has defined size limits to assist in determining when other objects are overlaid with the container. See Tables B.7 and B.8.

a) **Relationships:**

— **may contain** output field, input field, output graphic, output shape, picture graphic, button, container (containing only objects on this list), object pointer (referencing only objects on this list);

— **may reference** macro;

— **may be contained by** data mask, alarm mask, key, container, button.

b) **Allowed commands:**

— Hide/Show Object;

— Change Child Location;

— Change Child Position;

— Change Size.

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	Draw child object at current location in background colour to erase it. Refresh container (to redraw child object or objects).	Change Child Position Response
On Change Size	Change Size command	Draw child objects at current location in background colour to erase them. Refresh container (to redraw child object or objects).	Change Size Response

Table B.8 — Container attributes and record format

Attribute name	AID	Type	size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=3	3	Object Type = Container
Width		Integer	2	0-65535	4-5	Maximum width of the container's area in pixels. Objects or portions of objects outside the defined area are clipped.
Height		Integer	2	0-65535	6-7	Maximum height of the container's area in pixels. Objects or portions of objects outside the defined area are clipped.
Hidden		Boolean	1	0 or 1	8	0 = False, 1 = True. Indicates whether or not this container and its child objects are hidden (not displayed). (True = Hidden)
Number of objects to follow		Integer	1	0-255	9	Number of objects to follow even if zero. Each of these objects is "contained" in this object. Each object consists of 6 bytes: two (2) for object ID and four (4) for location.
Number of macros to follow		Integer	1	0-255	10	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Object ID}		Integer	2	0-65534	11+object*6	Object ID of an object contained in this container. List all objects before listing macros.
{X Location}		Signed integer	2	– 32767 to + 32767	13+object*6	Relative X location of the top left corner of the object (relative to the top left corner the container object).
{Y Location}		Signed integer	2	– 32767 to + 32767	15+object*6	Relative Y location of the top left corner of the object (relative to the top left corner of the container object).
Repeat: {Event ID}		Integer	1	0-255	11+(No. objects *6)…	(List these after all objects have been listed.) Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	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 virtual 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) **Relationships:**

— **may contain** key, object pointer (pointing to a key object);

— **may reference** macro;

— **may be contained by** none (top level).

b) **Allowed commands:**

— Change Background Colour;

— Change Attribute.

Table B.9 — Soft key mask events

Event	Caused by	VT behaviour	Message
On Show	Show parent alarm/data mask or change soft key mask command	Draw child objects in the order they are listed in the soft key mask object.	—
On Hide	Hide on parent alarm/data mask or change soft key mask command	—	—
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 above.	Change Attribute Response

Table B.10 — Soft key mask attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=4	3	Object Type = Soft Key Mask
Background colour	1	Integer	1	0-255	4	Background colour. The key object has its own background colour attribute that overrides this attribute.
Number of objects to follow		Integer	1	0-255	5	Number of objects to follow even if zero. Each of these objects is “contained” in this soft key mask.
Number of macros to follow		Integer	1	0-255	6	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Object ID}		Integer	2	0-65534	7+object* 2…	Object ID of a key or object pointer contained in this mask.
Repeat: {Event ID}		Integer	1	0-255	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	0-255	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.

a) **Relationships:**

— **may contain** output field, output shape, picture graphic, container (containing only objects on this list), object pointer (referencing only objects on this list);

— **may reference** macro;

— **may be contained by** soft key mask.

b) **Allowed commands:**

— Change Background Colour;

— Change Child Location;

— Change Child Position;

— Change Attribute.

Table B.11 — Key events

Event	Caused by	VT behaviour	Message
On Key Press	Operator pressing the soft key	—	Soft Key Activation
On Key Release	Operator releasing the soft key	—	Soft Key Activation
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	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.12 — Key attributes and record format

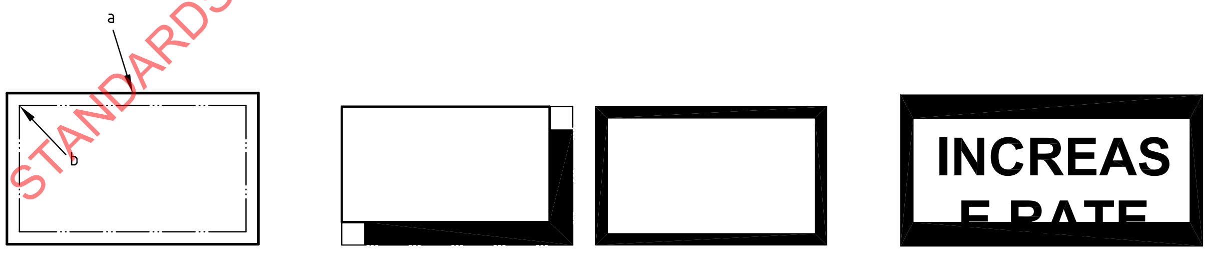
Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=5	3	Object type = key
Background colour	1	Integer	1	0-255	4	Background colour.
Key code	2	Integer	1	1-255	5	Key code assigned by ECU. VT reports this code in the soft key activation message. Note that key code zero (0) is reserved for use for the ACK means.
Number of objects to follow		Integer	1	0-255	6	Number of objects to follow even if zero. Each of these objects is “contained” in this object. Each object consists of 6 bytes: two (2) for object id and four (4) for location.

Table B.12 (*continued*)

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record byte**	**Description**
Number of macros to follow		Integer	1	0-255	7	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Object ID}		Integer	2	0-65534	8+object*6	Object ID of an object contained in this key. List all objects before listing macros.
{X Location}		Signed integer	2	– 32767 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 the soft key designator).
{Y Location}		Signed integer	2	– 32767 to + 32767	12+object*6	Relative Y location of the top left corner of the object in VT pixels (relative to the top left corner of the soft key designator).
Repeat: {Event ID}		Integer	1	0-255	8+(No. objects *6)…	(List these after all objects have been listed.) Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	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 pointing devices but shall be supported by all VTs. Alternatively, if a touch screen or pointing device is not supported, the VT shall provide a means for navigating to the button object (or objects) with the tab order being defined by the order in which the button objects were included in the parent mask. The working set can determine if the VT supports touch screen or pointing devices by using a Get Hardware command. When the button object is activated, the VT sends a Button Activation message to the owner working set master. A border of four pixels on all sides shall be reserved and the VT shall indicate when a button is selected, pressed or latched. The actual appearance of the border is proprietary to the VT design. Items outside the inner border are clipped as shown in Figure B.1. See Tables B.13 and B.14.



a) **Four pixels on all sides are reserved for border space**

b) **Button size cannot exceed size of enclosing rectangle regardless of proprietary appearance**

c) **Objects that do not fit the inner area are clipped**

[a] Area is defined by width and height attributes.

[b] Contained object positions are relative to top left inner corner.

Figure B.1 — Button examples

a) **Relationships:**

- **may contain** output field, output shape, picture graphic, container (containing only objects on this list), object pointer (referencing only objects on this list);
- **may reference** macro;
- **may be contained by** data mask, container.

b) **Allowed commands:**

- Change Background Colour;
- Change Size;
- Change Child Location;
- Change Child Position;
- Change Attribute.

Table B.13 — Button events

Event	Caused by	VT behaviour	Message
On Key Press	Operator activating the button	—	Button Activation
On Key Release	Operator releasing an 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	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	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=6	3	Object Type = Button
Width	1	Integer	2	0-65535	4-5	Maximum width of the button's area in pixels.
Height	2	Integer	2	0-65535	6-7	Maximum height of the button's area in pixels.
Background colour	3	Integer	1	0-255	8	Background colour.
Border colour	4	Integer	1	0-255	9	Border colour.

Table B.14 (*continued*)

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record byte**	**Description**
Key Code	5	Integer	1	0-255	10	Key code assigned by ECU. VT reports this code in the button activation message.
Latchable		Boolean	1	0,1 or 3	11	A value of 0 indicates that the button is not latchable or is momentary. A value of 1 indicates that the button is latchable and its initial state is not latched. A value of 3 indicates that the button is latchable and that its initial state is latched.
Number of objects to follow		Integer	1	0-255	12	Number of objects to follow even if zero. Each of these objects is “contained” in this object. Each object consists of 6 bytes: two (2) for object ID and four (4) for location.
Number of macros to follow		Integer	1	0-255	13	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Object ID}		Integer	2	0-65534	14+object* 6	Object ID of an object contained in this button. List all objects before listing macros.
{X Location}		Signed integer	2	– 32767 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 the button). Objects or portions of objects outside the inner border are clipped.
{Y Location}		Signed integer	2	– 32767 to + 32767	18+object* 6	Relative Y location of the top left corner of the object in VT pixels (relative to the top left inner corner of the button). Objects or portions of objects outside the inner border are clipped.
Repeat: {Event ID}		Integer	1	0-255	14+(No. objects *6)…	(List these after all objects have been listed.) Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	15+(No. objects *6)…	Macro ID of the macro to execute.

B.8 Input field objects

B.8.1 General

There are four types of input field: Boolean, string, number, and list. Input Boolean, input string and input number 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), and shown and disabled (displayed but not able to accept input). In order to hide an input field object, it shall be contained by a container object. See Tables B.15 to B.20.

a) **Relationships (does not apply to input list object):**

- **may contain** none (atomic);
- **may reference** macro, variable, font attributes, input attributes;
- **may be contained by** data mask, container.

b) **Allowed commands (does not apply to input list object):**

- Enable/Disable Object;
- Select Input Object;
- ESC;
- Change Background Colour;
- Change Numeric Value;
- Change String Value (Input String object only);
- Change Attribute;
- Change Size.

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 command	Mark the input object enabled. If displayed, operator may navigate to it.	Enable/Disable Object Response
On Disable	Enable/Disable 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	Operator navigates to input field	The VT shall provide some way for the operator to recognize that the input field is selected for input.	VT Select Input Object
On input Field De-selection	Operator navigates off input field or as a result of a disable event	—	VT Select Input Object
On ESC	Operator aborts input using ESC key or working set sends an ESC message	Revert value of object to value before operator began the edit. Redraw this object. Refresh parent object.	ESC Response message on ESC message or VTESC message if operator activated the ESC means
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 Numeric Value	Change Numeric Value or Change String Value command	If input object is displayed, redraw object with new value. Refresh parent object.	Change Value Response (for Change Numeric Value only)
On Entry of Value	Operator saving changes by use of the ENTER means regardless of whether or not the value changed	VT Change Numeric	VT Change Numeric Value or Input String Value
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

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

Table B.16 — Input Boolean attributes and record format

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record byte**	**Description**
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=7	3	Object Type = Input Boolean
Background colour	1	Integer	1	0-255	4	Background colour.
Width	2	Integer	2	0-65535	5-6	Maximum width and height of the input field in pixels.
Foreground colour	3	Integer	2	0-65534	7-8	Object ID of a font attributes object to use for display formatting of this field. The only useful attribute is the font colour
Variable reference	4	Integer	2	0-65534, 65535	9-10	Object ID of a number variable object in which to store or retrieve the object's value. If this attribute is set to NULL (65535), the value is stored directly in the value attribute instead.
Value		Integer	1	0 or 1	11	Value of the input field. 0 for FALSE or 1 for TRUE. Used only if variable reference attribute is NULL (65535).
Enabled		Integer	1	0 or 1	12	Initial state of object. 0 = Disabled, 1 = Enabled
Number of macros to follow		Integer	1	0-255	13	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Event ID}		Integer	1	0-255	14…	Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	15…	Macro ID of the macro to execute.

B.8.3 Input string

This object is used to input a character string from the operator. Displayable characters are shown in Table L.1. Several special formatting characters are permitted in the input string's value and shall be properly interpreted by the VT, as follows.

— Line Feed (Code 10_{10}): moves the text cursor down one line at the equal × position based on font size and text area dimensions. The line feed character code is entered into the string's value.

— Carriage Return (Code 13_{10}): moves the text cursor to the left edge of the current line. This will be the × coordinate position specified by the text area dimensions. The carriage return character code is entered into the string's value.

The VT design may allow both codes to be entered with a single key press.

Table B.17 — Input string attributes and record format

Attribute name	AID	Type	size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=8	3	Object Type = Input String
Width	1	Integer	2	0-65535	4-5	Maximum width of the input field in pixels. Objects or portions of objects outside the defined area are clipped.
Height	2	Integer	2	0-65535	6-7	Maximum height of the input field in pixels. Objects or portions of objects outside the defined area are clipped.
Background colour	3	Integer	1	0-255	8	Background colour. Used only if the "transparent" bit in the options attribute is cleared. To be applied in the complete rectangle specified by Width and Height.
Font attributes	4	Integer	2	0-65534	9-10	Object ID of a font attributes object to use for display formatting of this field.
Input attributes	5	Integer	2	0-65534, 65535	11-12	Object ID of an input attributes object to use for character string validation or NULL (65535) for no validation.
Options	6	Bitmask	1	0-3	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 and the amount of text to display is longer than the width, the VT will format and wrap the text into the next line(s). The text may not exceed the boundaries defined by the width and height.
Variable reference	7	Integer	2	0-65534, 65535	14-15	Object ID of a string variable object in which to store or retrieve the object's value. If this attribute is set to NULL (65535), the value is stored directly in the value attribute instead. **IMPORTANT — If this attribute is < 65535, the Length attribute shall be set to zero and the value attribute is EXCLUDED from this record.**
Horizontal justification	8	Integer	1	0-2	16	Field justification. Indicates how the text string is positioned horizontally within the field defined by width and height. This is a character justification, not a pixel justification. The VT shall add leading space characters to perform the justification. 0 = Position Left 1 = Position Middle 2 = Position Right Vertical justification is not supported. During editing of this object, the VT designer may choose to suppress justification until the field is deselected.

Table B.17 (*continued*)

Attribute name	**AID**	**Type**	**size** bytes	**Range or value**	**Record byte**	**Description**
Length		Integer	1	0-255	17	Maximum fixed length of the input string value in characters. Set to zero if a variable reference is used. When variable reference is used, its variable must not exceed 255, since the length attribute is only one byte.
Value		String	Length		18…	Value of the input field based on current character set. Used only if length is non-zero (0) and variable reference attribute is NULL (65535). Pad with spaces as necessary to satisfy length attribute.
Enabled		Integer	1	0 or 1	Depends on size of value attribute	Initial state of object. 0 = Disabled, 1 = Enabled
Number of macros to follow		Integer	1	0-255	Depends on size of value attribute	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Event ID}		Integer	1	0-255	Depends on size of value attribute	Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	Depends on size of value attribute	Macro ID of the macro to execute.

B.8.4 Input number

This object is used to input a 32-bit integer or float. The stored and transmitted value is always a 32-bit unsigned integer but the displayed value is scaled and formatted. The following equations shall be true:

Displayed value = (value attribute + Offset) * Scaling Factor
Transmitted value (VT to working set) = (Displayed value / Scaling Factor) – Offset = Value Attribute
Scaled Upper Limit = (Upper Limit + Offset) * Scaling Factor
Scaled Lower Limit = (Lower Limit + Offset) * Scaling Factor
Lower Limit <= value attribute <= Upper Limit
Scaled Lower Limit <= Displayed value <= Scaled Upper Limit

Table B.18 — Input number attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=9	3	Object Type = input Number
Width	1	Integer	2	0-65535	4-5	Maximum width of the input field in pixels. Objects or portions of objects outside the defined area are clipped.
Height	2	Integer	2	0-65535	6-7	Maximum height of the input field in pixels. Objects or portions of objects outside the defined area are clipped.
Background colour	3	Integer	1	0-255	8	Background colour. Used only if the “transparent” bit in the options attribute is cleared. To be applied in the complete rectangle specified by Width and Height.
Font attributes	4	Integer	2	0-65534	9-10	Object ID of a font attributes object to use for display formatting of this field.
Options	5	Bitmask	1	0-7	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.
Variable reference	6	Integer	2	0-65534, 65535	12-13	Object ID of a number variable object in which to store or retrieve the object’s raw unscaled value. If this attribute is set to NULL (65535), the value is stored directly in the value attribute instead. VT transmits the raw unscaled value to the working set.
Value		Integer	4	0-2^32	14-17	Raw unsigned value of the input field before scaling (unsigned 32-bit integer). Used only if variable reference attribute is NULL (65535). VT transmits the raw unscaled value to the working set.
Min value	7	Integer	4	0-2^32	18-21	Raw minimum value for the input before scaling. Offset and scaling shall be applied to determine the actual minimum value.
Max value	8	Integer	4	0-2^32	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	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	0-7	34	Specifies number of decimals to display after the decimal point.

Table B.18 (*continued*)

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record byte**	**Description**
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.
Horizontal justification	13	Integer	1	0-2	36	Field justification. Indicates how the number is positioned horizontally within the field defined by width and height. This is a character justification, not a pixel justification. The VT shall add leading space characters to perform the justification. 0 = Position Left 1 = Position Middle 2 = Position Right Vertical justification is not supported. During editing of this object, the VT designer may choose to suppress justification until the field is deselected.
Enabled		Integer	1	0 or 1	37	Initial state of object. 0 = Disabled, 1 = Enabled
Number of macros to follow		Integer	1	0-255	38	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Event ID}		Integer	1	0-255	39…	Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	40…	Macro ID of the macro to execute.

B.8.5 Input list

IMPORTANT — 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 might be to display values as the operator moves through the list using +/– keys. A more complex implementation might 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 selected. 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 output field or picture graphic objects. The value transmitted to the working set master is the list index chosen (range 0-255).

a) **Relationships:**

— **may contain** output field, picture graphic;

— **may reference** macro, variable;

— **may be contained by** data mask, container.

b) **Allowed commands:**

— Enable/Disable Object;

— Select Input Object;

— ESC;

— Change Numeric Value;

— Change Attribute;

— Change List Item;

— Change Size.

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 command	Mark the input object enabled. If displayed, operator may navigate to it.	Enable/Disable Object Response
On Disable	Enable/Disable 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	Operator navigates to input object	The VT shall provide some way for the operator to recognize that the input object is selected for input.	VT Select Input Object
On input Field De-selection	Operator navigates off input object or as a result of a disable event	—	VT Select Input Object
On ESC	Operator aborts input using ESC key or working set sends an ESC message	Revert value of object to value before operator began the edit. Redraw this object. Refresh parent object.	ESC response message on ESC message or VT ESC message if operator activated the ESC means
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 Change Numeric	VT Change Numeric Value or Input String Value
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 — Attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=10	3	Object Type = input List
Width	1	Integer	2	0-65535	4-5	Maximum width of the input field in pixels. Objects or portions of objects outside the defined area are clipped.
Height	2	Integer	2	0-65535	6-7	Maximum height of the input field in pixels. Objects or portions of objects outside the defined area are clipped.
Variable reference	3	Integer	2	0-65534, 65535	8-9	Object ID of a number variable object in which to store or retrieve the object's value. If this attribute is set to NULL (65535), the value is stored directly in the value attribute instead.
Value		Integer	1	0-254, 255	10	Selected list index of this object. Used only if variable reference attribute is NULL (65535). The current list item chosen or 255 to indicate no item is chosen. The first item is at index zero (0).
Number of list items		Integer	1	0-255	11	Number of object references to follow. The size of the list can never exceed this number and this attribute cannot be changed.
Enabled		Integer	1	0 or 1	12	Initial state of object. 0 = Disabled, 1 = Enabled
Number of macros to follow		Integer	1	0-255	13	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {object ID}		Integer	2	0-65534, 65535	14+object* 2	These objects make up the list. Object ID of an output field or picture graphic object. Null (65535) is a no-item placeholder (empty object). The VT shall skip empty items. The change list item command allows objects to be replaced or removed.
Repeat: {Event ID}		Integer	1	0-255	14+(No. List Items * 2)…	(List these after all objects have been listed.) Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	15+(No. List Items * 2)…	Macro ID of the macro to execute.

B.9 Output field objects

B.9.1 General

There are two types of output field: string and number. They have similar relationships and behaviour, but have different attributes. See Tables B.21 to B.23.

a) **Relationships:**

— **may contain** none (atomic);

— **may reference** macro, variable, font attributes;

— **may be contained by** data mask, alarm mask, key, container, input list, button, auxiliary function, auxiliary input.

b) **Allowed commands:**

— Change Background Colour;

— Change Numeric Value;

— Change String Value (Output String object only);

— Change Attribute;

— Change Size.

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 or Change String Value command	If output object is displayed, redraw object with new value. Refresh parent object.	Change Value Response (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

This object is used to output a string of text. Displayable characters are given in Table L.1. Several special formatting characters are permitted in the output string's value and shall be properly interpreted by the VT as follows.

— Backspace (non-destructive) (Code 8_{10}): moves the text cursor back one font character width left but not farther to the left than the × coordinate position specified by the text area dimensions.

— Line feed (Code 10_{10}): moves the text cursor down one line at the equal × position based on font size and text area dimensions.

— Carriage return (Code 13_{10}): moves the text cursor to the left edge of the current line. This will be the × coordinate position specified by the text area dimensions.

Table B.22 — Output string attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=11	3	Object Type = Output String
Width	1	Integer	2	0-65535	4-5	Maximum width of the output field in pixels. Objects or portions of objects outside the defined area are clipped.
Height	2	Integer	2	0-65535	6-7	Maximum height of the output field in pixels. Objects or portions of objects outside the defined area are clipped.
Background colour	3	Integer	1	0-255	8	Background colour. Used only if the "transparent" bit in the options attribute is cleared. To be applied in the complete rectangle specified by Width and Height.
Font attributes	4	Integer	2	0-65534	9-10	Object ID of a font attributes object to use for display formatting of this field.
Options	5	Bitmask	1	0-3	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 and the amount of text to display is longer than the width, the VT will format and wrap the text into the next line(s). The text may not exceed the boundaries defined by the width and height. Wrap on space between words. If there is no space in the line, then wrap on the last character in the line.
Variable reference	6	Integer	2	0-65534, 65535	12-13	Object ID of a string variable object from which to retrieve the object's value. If this attribute is set to NULL (65535), the string is stored directly in the value attribute instead. **IMPORTANT — If this attribute is < 65535, the length attribute shall be set to zero and the value attribute is *excluded* from this record.**
Horizontal justification	7	Integer	1	0-2	14	Field justification. Indicates how the text string is positioned horizontally within the field defined by width and height. This is a character justification, not a pixel justification. The VT shall add leading space characters to perform the justification. 0 = Position Left 1 = Position Middle 2 = Position Right Vertical justification is not supported.
Length		Integer	2	0-65535	15-16	Maximum fixed length of the output string value in characters. Set to zero if a variable reference is used.

Table B.22 (*continued*)

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Value		String	Length		17-n	Text string to output in the output field. If Length is zero, this attribute is excluded from the record. Pad with spaces as necessary to satisfy length attribute. May also contain formatting codes as described above.
Number of macros to follow		Integer	1	0-255	Depends on size of string	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Event ID}		Integer	1	0-255	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	0-255	Depends on size of string	Macro ID of the macro to execute.

B.9.3 Output number

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

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

Displayed values are always rounded to the number of decimals specified in the decimal attribute defined in Table B.23.

Table B.23 — Output number attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=12	3	Object Type = Output Number
Width	1	Integer	2	0-65535	4-5	Maximum width of the output field in pixels. Objects or portions of objects outside the defined area are clipped.
Height	2	Integer	2	0-65535	6-7	Maximum height of the output field in pixels. Objects or portions of objects outside the defined area are clipped.
Background colour	3	Integer	1	0-255	8	Background colour. Used only if the “transparent” bit in the options attribute is cleared. To be applied in the complete rectangle specified by Width and Height.
Font attributes	4	Integer	2	0-65534	9-10	Object ID of a font attributes object to use for display formatting of this field.

Table B.23 (*continued*)

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record byte**	**Description**
Options	5	Bitmask	1	0-7	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 = 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.
Variable reference	6	Integer	2	0-65534, 65535	12-13	Object ID of an integer variable object in which to retrieve the object's raw unscaled value. If this attribute is set to NULL (65535), the value is retrieved directly from the value attribute instead. VT shall scale the value for display.
Value		Integer	4	0-2^32	14-17	Raw unsigned value of the output field before scaling (unsigned 32-bit integer). Used only if variable reference attribute is NULL (65535). VT shall scale this value for display.
Offset	7	Signed Integer	4	-2^31 to 2^31	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	0-7	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).
Horizontal justification	11	Integer	1	0-2	28	Field justification. Indicates how the number is positioned horizontally within the field defined by width and height. This is a character justification, not a pixel justification. The VT shall add leading space characters to perform the justification. 0 = Position Left 1 = Position Middle 2 = Position Right Vertical justification is not supported.
Number of macros to follow		Integer	1	0-255	29	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Event ID}		Integer	1	0-255	30…	Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	31…	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 Figures B.2 to B.5 and Tables B.24 to B.31.

B.10.2 Line

This function draws a line object. The starting point for the line is found in the parent object.

a) **Relationships:**

— **may contain** none (atomic);

— **may reference** macro, line attributes;

— **may be contained by** data mask, alarm mask, key, button, container, auxiliary function, auxiliary input.

b) **Commands:**

— Change End Point;

— Change Attribute;

— Change Size.

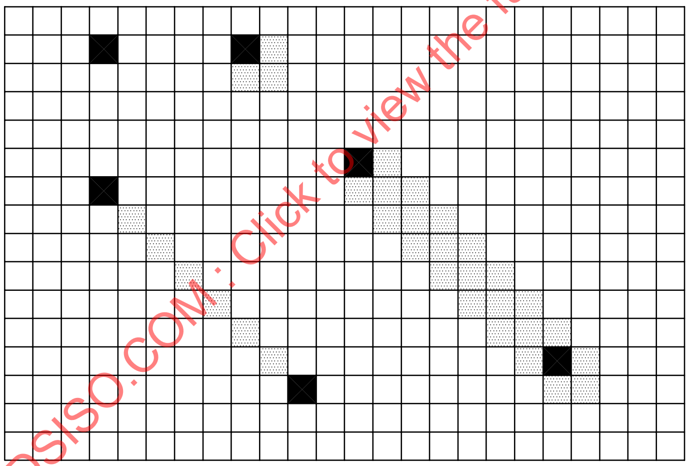


Figure B.2 — Line object showing start and end points using different brush sizes

Table B.24 — 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.25 — Line attributes and record format

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record byte**	**Description**
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=13	3	Object Type = Line
Line attributes	1	Integer	2	0-65534	4-5	Object ID of a line attributes object to use for the line attributes.
Width	2	Integer	2	0-65535	6-7	Width in pixels of an enclosing virtual rectangle. (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.2 for an example of start and end points and line width.
Height	3	Integer	2	0-65535	8-9	Height in pixels of an enclosing virtual rectangle. (StartX, StartY) to (StartX + Width – 1, StartY + Height – 1) *inclusive* defines the graphical clipping limits when drawing this object. Endpoint can be calculated as follows: — EndPointX = StartX + Width – LineWidth — EndPointY = StartY + Height – LineWidth See Figure B.2 for an example of start and end points and line width.
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	0-255	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	0-255	12…	Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	13…	Macro ID of the macro to execute.

B.10.3 Rectangle object

See Figure B.3.

a) **Relationships:**

— **may contain** none (atomic);

— **may reference** macro, line attributes, fill attributes;

— **may be contained by** data mask, alarm mask, key, button, container, auxiliary function, auxiliary input.

b) **Allowed commands:**

— Change Size;

— Change Attribute.

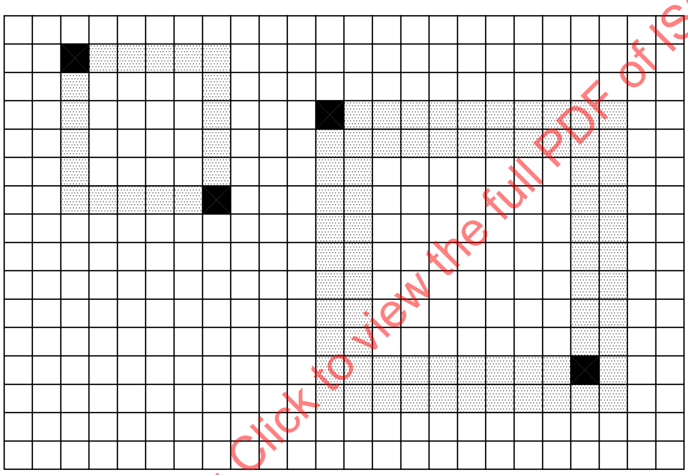


Figure B.3 — Rectangle object showing end points using different brush sizes

Table B.26 — Output rectangle events

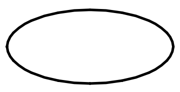
Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Redraw this object.	—
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size Response
On Change Attribute	Change Attribute command	Redraw this object, refresh parent mask.	Change Attribute Response

Table B.27 — Output rectangle attributes and record format

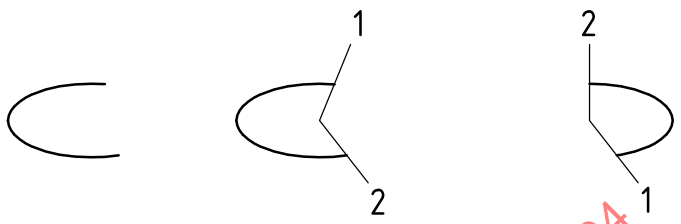
Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record byte**	**Description**
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=14	3	Object Type = Rectangle
Line attributes	1	Integer	2	0-65534	4-5	Object ID of a line attributes object to use for the line attributes.
Width	2	Integer	2	0-65535	6-7	Width in pixels. (StartX, StartY) to (StartX + Width – 1, StartY + Height – 1) *inclusive* defines the graphical clipping limits when drawing this object. Endpoint can be calculated as follows: — EndPointX = StartX + Width – LineWidth — EndPointY = StartY + Height – LineWidth See Figure B.3 for an example of start and end points and line width.
Height	3	Integer	2	0-65535	8-9	Height in pixels. (StartX, StartY) to (StartX + Width – 1, StartY + Height – 1) *inclusive* defines the graphical clipping limits when drawing this object. Endpoint can be calculated as follows: — EndPointX = StartX + Width – LineWidth — EndPointY = StartY + Height – LineWidth See Figure B.3 for an example of start and end points and line width.
Line suppression	4	Bitmask	1	0-15	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) NOTE When drawing a filled rectangle with line suppression, only the pixels that would be on the border of the rectangle are suppressed (not drawn). Line width shall be taken into account to know the width of the border.
Fill attributes	5	Integer	2	0-65534, 65535	11-12	Object ID of a fill attributes object to use for the fill attributes or NULL (65535) for no fill.
Number of macros to follow		Integer	1	0-255	13	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Event ID}		Integer	1	0-255	14…	Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	15…	Macro ID of the macro to execute.

B.10.4 Ellipse

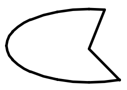
This object outputs an ellipse or circle shape. Several options are available for modifying the appearance (see Figure B.4).



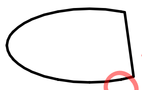
a) **Type = 0: Closed ellipse (fillable)**



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.4 — Ellipse object

a) **Relationships:**

- **may contain** none (atomic);
- **may reference** macro, line attributes, fill attributes;
- **may be contained by** data mask, alarm mask, key, button, container, auxiliary function, auxiliary input.

b) **Allowed commands:**

- Change Size;
- Change Attribute.

Table B.28 — 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.29 — Output ellipse attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=15	3	Object Type = Ellipse
Line attributes	1	Integer	2	0-65534	4-5	Object ID of a line attributes object to use for the line attributes.
Width	2	Integer	2	0-65535	6-7	Width in pixels of an enclosing virtual rectangle. (StartX, StartY) to (StartX + Width – 1, StartY + Height – 1) *inclusive* defines the graphical clipping limits when drawing this object.
Height	3	Integer	2	0-65535	8-9	Height in pixels of an enclosing virtual rectangle. (StartX, StartY) to (StartX + Width – 1, StartY + Height – 1) *inclusive* defines the graphical clipping limits when drawing this object.
Ellipse type	4	Integer	1	0-3	10	Type of ellipse: 0 = Closed Ellipse 1 = Open Ellipse defined by start/end angles 2 = Closed Ellipse Segment (see Figure B.4) 3 = Closed Ellipse Section (see Figure B.4) NOTE If type > 0 and start and end angles are the same, the ellipse is drawn closed.
Start angle	5	Integer	1	0-180	11	Start angle/2 (in degrees) from positive X axis counter clockwise (90° is straight up). Start and end angles define the arc.
End angle	6	Integer	1	0-180	12	End angle/2 (in degrees) from positive X axis counter clockwise (90° is straight up). Start and end angles define the arc.
Fill attributes	7	Integer	2	0-65534, 65535	13-14	Object ID of a Fill attributes object to use for the fill attributes or NULL (65535) for no fill.
Number of macros to follow		Integer	1	0-255	15	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Event ID}		Integer	1	0-255	16…	Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	17…	Macro ID of the macro to execute.

B.10.5 Polygon

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.

The start point for the polygon is the first point listed. Polygon is drawn using the points in the list in the order given. At least three points are required for a polygon. The point positions are relative to the upper left-hand corner of the polygon object and the upper left-hand 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.

See Figure B.5.

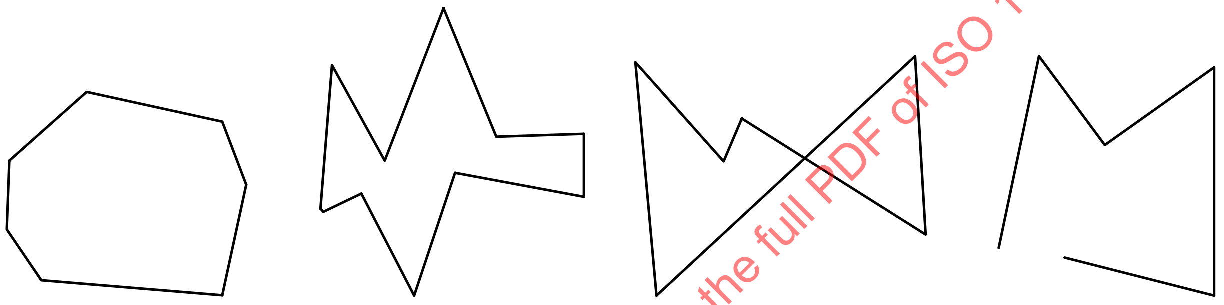


Figure B.5 — Polygon types

a) **Relationships:**

— **may contain** none (atomic);

— **may reference** macro, line attributes, fill attributes;

— **may be contained by** data mask, alarm mask, key, button, container, auxiliary function, auxiliary input.

b) **Allowed commands:**

— Change Attribute;

— Change Size.

Table B.30 — 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.31 — Output polygon attributes and record format

Attribute name	**AID**	**Type**	**Size** bytes	**Range or value**	**Record byte**	**Description**
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=16	3	Object Type = Polygon
Width	1	Integer	2	0-65535	4-5	Width in pixels of an enclosing virtual rectangle. (StartX, StartY) to (StartX + Width – 1, StartY + Height – 1) *inclusive* defines the graphical clipping limits when drawing this object.
Height	2	Integer	2	0-65535	6-7	Height in pixels of an enclosing virtual rectangle. (StartX, StartY) to (StartX + Width – 1, StartY + Height – 1) *inclusive* defines the graphical clipping limits when drawing this object.
Line attributes	3	Integer	2	0-65534	8-9	Object ID of a line attributes object to use for the line attributes.
Fill attributes	4	Integer	2	0-65534, 65535	10-11	Object ID of a Fill attributes object to use for the fill attributes or NULL (65535) for no fill.
Polygon type	5	Integer	1	0-3	12	Polygon type. The first three types are useful only if the polygon is to be filled. VT designer 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 may be encountered but the polygon edges do not cross. 2 = Complex. Similar to Non-convex but edges cross. 3 = Open. This type cannot be filled.
Number of points		Integer	1	3-255	13	Number of points to follow. Each point is 4 bytes. At least three (3) points shall be listed or this object cannot exist.
Number of macros to follow		Integer	1	0-255	14	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Point X}		Integer	2	0-65535	15+point#* 4	X value of a point relative to the top left corner of the polygon.
{Point Y}		Integer	2	0-65535	17+point#* 4	Y value of a point relative to the top left corner of the polygon.
Repeat: {Event ID}		Integer	1	0-255	15+(num points * 4)…	(List macros after all points.) Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	16+(num points * 4)…	Macro ID of the macro to execute.
* Indicates multiplication.						

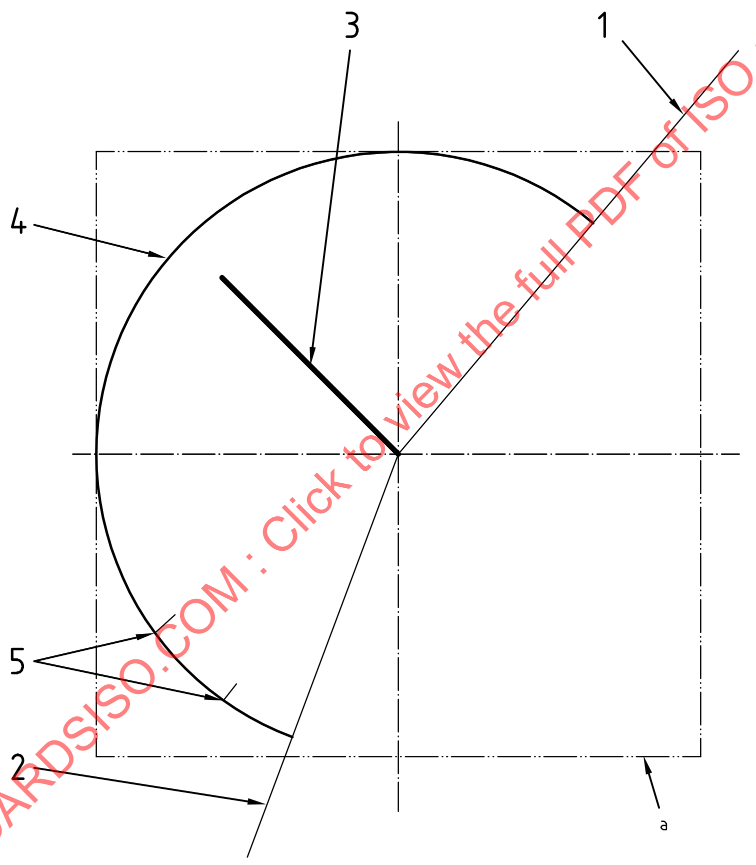
B.11 Output graphic objects

B.11.1 General

There are three types of output graphic object: meter, linear bar graph and arched bar graph.

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.6 and B.7 and Tables B.32 and B.33.



Key

1 start angle
2 end angle
3 needle
4 arc
5 ticks

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

Figure B.6 — Meter object

a) **Relationships:**

— **may contain** none (atomic);

— **may reference** macro, variable;

— **may be contained by** data mask, alarm mask, container.

b) **Allowed commands:**

— Change Numeric Value;

— Change Attribute;

— Change Size.

Table B.32 — Output meter events

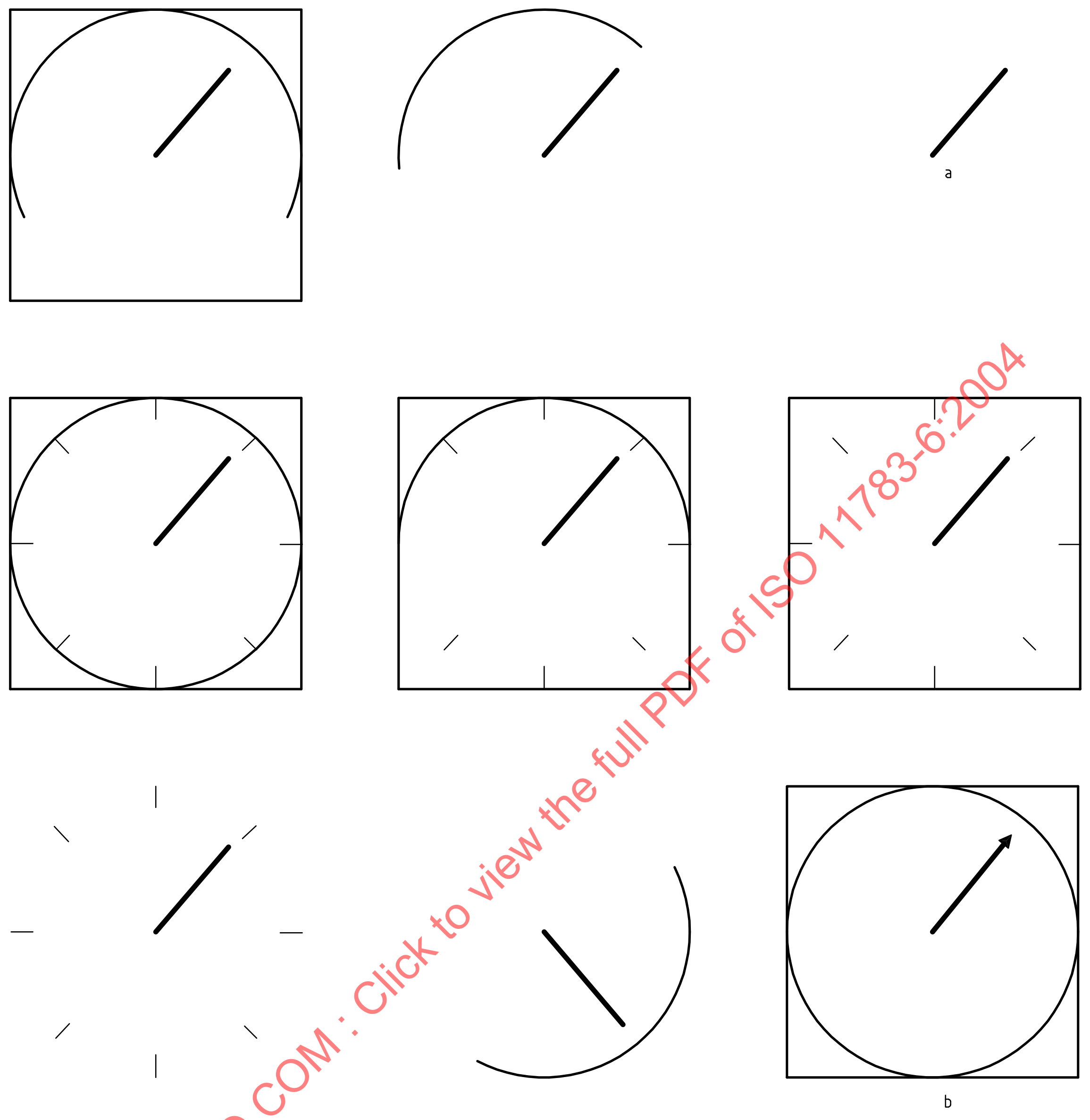
Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Redraw this object.	—
On Change Value	Change Numeric Value command	Redraw this object, refresh parent mask.	Change Numeric Value Response
On Change Attribute	Change Attribute command	Redraw this object, refresh parent mask.	Change Attribute Response
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size Response

Table B.33 — Output meter attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=17	3	Object Type = Output Meter
Width	1	Integer	2	0-65535	4-5	Maximum width and height of the enclosing square in pixels. Meter object cannot exceed the bounds of this imaginary square.
Needle colour	2	Integer	1	0-255	6	Needle (indicator) colour.
Border colour	3	Integer	1	0-255	7	Border colour (if drawn).
Arc and tick colour	4	Integer	1	0-255	8	meter arc and tick colour (if drawn).
Options	5	Bitmask	1	0-15	9	Logical bits to indicate options. 1 = True. Bit 0 = Draw Arc Bit 1 = Draw Border Bit 2 = Draw Ticks Bit 3 = Deflection Direction. 0 = From minimum to maximum, counter clockwise. 1 = From minimum to maximum, clockwise

Table B.33 (*continued*)

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
# of ticks	6	Integer	1	0-255	10	Number of ticks to draw about meter arc. If one tick, it is drawn in the middle of the arc. For two or more ticks a tick is placed at each end of the arc and the rest are evenly spaced between them.
Start angle	7	Integer	1	0-180	11	Start angle/2 (in degrees) from positive X axis anticlockwise (90° is straight up). Start and end angles define the arc. If the start and end angles are the same the meter's arc is closed (360°).
End angle	8	Integer	1	0-180	12	End angle/2 (in degrees) from positive X axis anticlockwise (90° is straight up). Start and end angles define the arc. If the start and end angles are the same the meter's arc is closed (360°).
Min value	9	Integer	2	0-65535	13-14	Minimum value. Represents value when needle is at start of arc.
Max value	10	Integer	2	0-65535	15-16	Maximum value. Represents value when needle is at end of arc.
Variable reference	11	Integer	2	0-65534, 65535	17-18	Object ID of a number variable object in which to retrieve the meter's value. If this attribute is set to NULL (65535), the value is retrieved directly from the value attribute instead. Number variable's value shall be in the range 0-65535.
Value		Integer	2	0-65535	19-20	Current value. Needle position is set by this value. Used only if variable reference is NULL (65535).
Number of macros to follow		Integer	1	0-255	21	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Event ID}		Integer	1	0-255	22…	Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	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.7 — 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.8 and Tables B.34 and B.35.

a) **Relationships:**

— **may contain** none (atomic);

— **may reference** macro, variable;

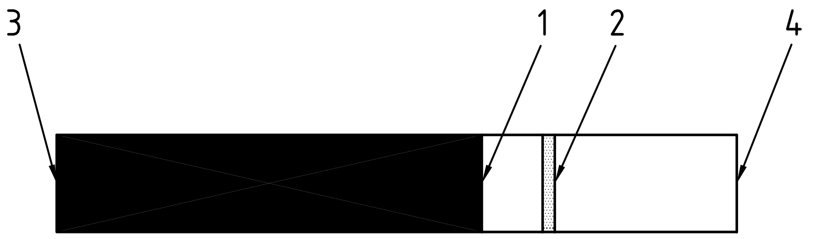
— **may be contained by** data mask, alarm mask, container.

b) **Allowed commands:**

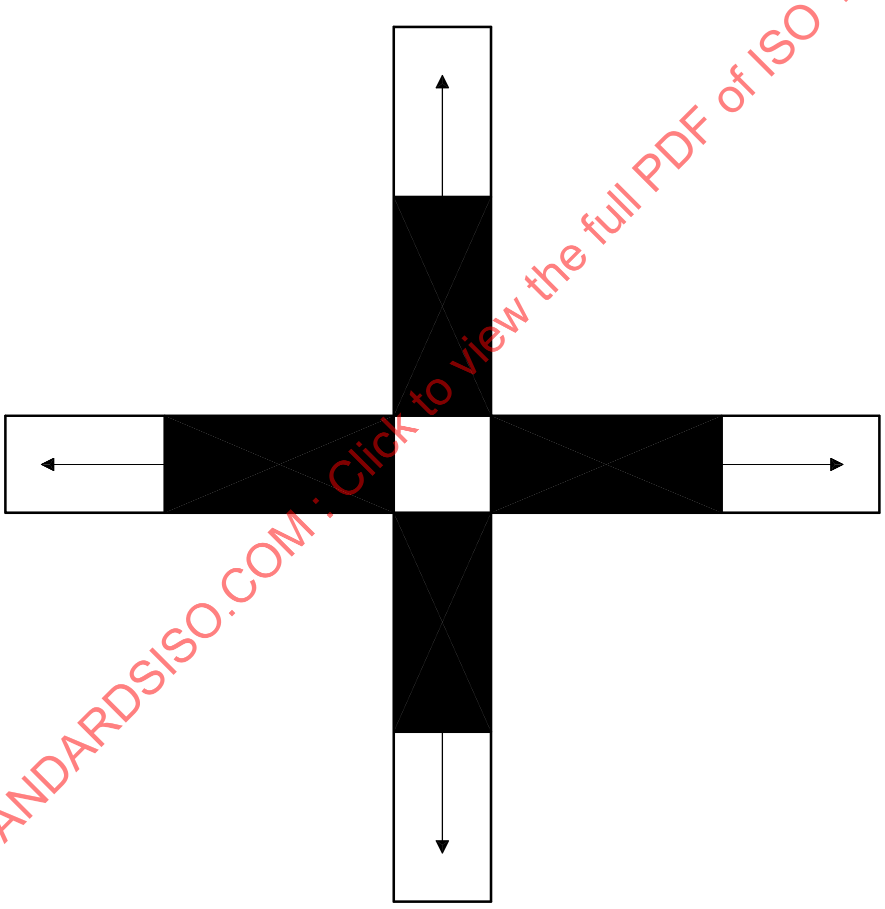
— Change Numeric Value;

— Change Attribute;

— Change Size.



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.8 — Linear bar graph — Examples

Table B.34 — Output linear bar graph events

Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Redraw this object.	—
On Change Value	Change Numeric Value command	Redraw this object, refresh parent mask.	Change Numeric Value Response
On Change Attribute	Change Attribute command	Redraw this object, refresh parent mask.	Change Attribute Response
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size Response

Table B.35 — Output linear bar graph attributes and record format

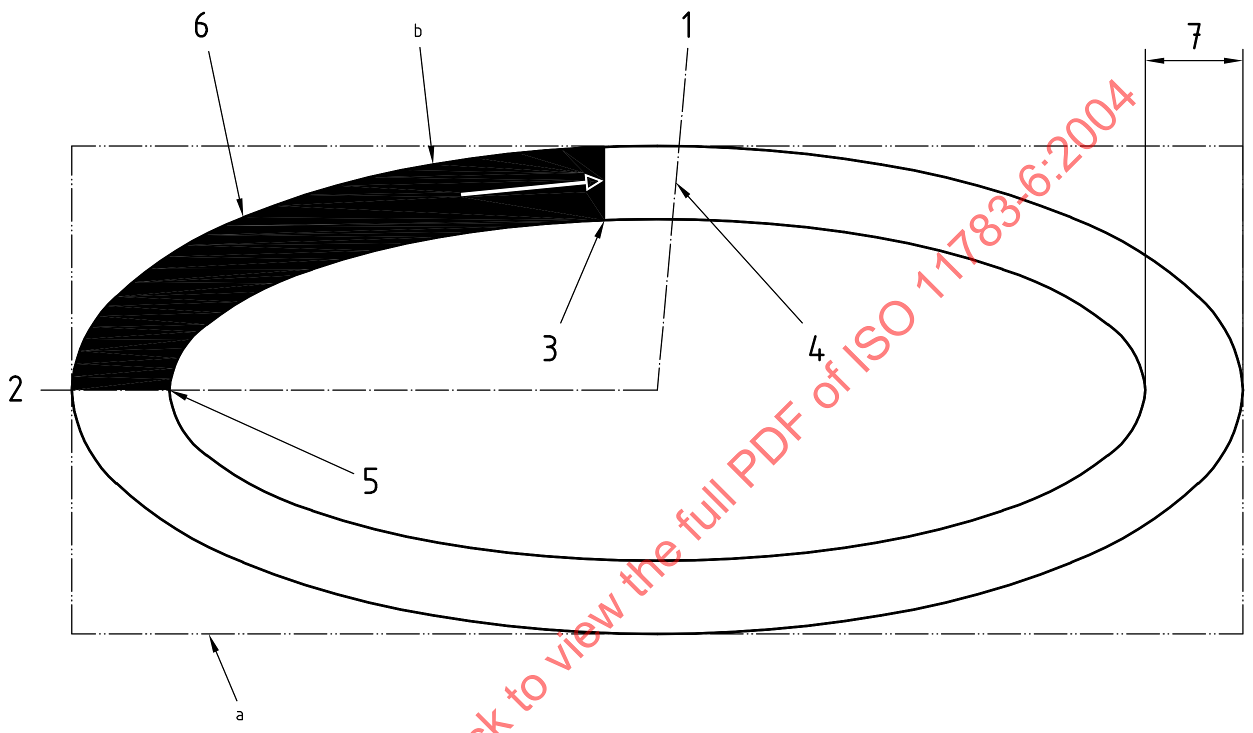
Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=18	3	Object Type = output linear bar graph
Width	1	Integer	2	0-65535	4-5	Maximum width of the enclosing rectangle in pixels. Bar graph cannot exceed the bounds of this imaginary rectangle.
Height	2	Integer	2	0-65535	6-7	Maximum height of the enclosing rectangle in pixels. Bar graph cannot exceed the bounds of this imaginary rectangle.
Colour	3	Integer	1	0-255	8	Bar graph fill and border colour.
Target line colour	4	Integer	1	0-255	9	Target line colour (if drawn).
Options	5	Bitmask	1	0-63	10	Logical bits to indicate which parts to draw: 1 = True Bit 0 = Draw border Bit 1 = Draw target line Bit 2 = Draw ticks Bit 3 = bar graph type. If this bit is FALSE (0), bar graph is filled. If this bit is TRUE (1), bar graph is not filled but rather shows the current value as a single line at the proper position within the bar graph. Orientation and direction of the bar graph: Bit 4 = Axis orientation. 0 = vertical (increasing values move parallel to the Y axis with constant X), 1 = horizontal (increasing values move parallel to the X axis with constant Y) Bit 5 = Direction. 0 = Grows negative (left or down). 1 = Grows positive (right or up).

Table B.35 (*continued*)

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
# of ticks	6	Integer	1	0-255	11	Number of ticks to draw along bar graph. If one tick, it is drawn in the middle of the bar graph. For two or more ticks a tick is placed at each end of the bar graph and the rest are evenly spaced between them.
Min value	7	Integer	2	0-65535	12-13	Minimum value.
Max value	8	Integer	2	0-65535	14-15	Maximum value.
Variable reference	9	Integer	2	0-65534, 65535	16-17	Object ID of a number variable object in which to retrieve the bar graph's value. If this attribute is set to NULL (65535), the value is retrieved directly from the value attribute instead. Number variable's value shall be in the range 0-65535.
Value		Integer	2	0-65535	18-19	Current value. Used only if variable reference is NULL (65535). Bar graph fills or moves, depending on bar graph type, to a point calculated from this value and min/max values. If value > Max value or value < Min value, the bar graph is filled or shown empty and no error is generated by the VT.
Target value variable reference	10	Integer	2	0-65534, 65535	20-21	Object ID of a number variable object in which to retrieve the bar graph's target value. If this attribute is set to NULL (65535), the target value is retrieved directly from the Target value attribute instead. Number variable's value shall be in the range 0-65535.
Target value	11	Integer	2	0-65535	22-23	Current target value. Used only if Target value variable Reference attribute is NULL (65535). Target value is displayed as a line on the bar graph to indicate some target or warning level. If Target value > Max value or Target value < Min value, the target line is shown on one of the ends of the bar graph and no error is generated by the VT.
Number of macros to follow		Integer	1	0-255	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	0-255	25…	Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	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-hand corner of the enclosing rectangle regardless of orientation. This object is drawn transparent so that objects can be placed underneath to enhance the appearance. See Figure B.9 and Tables B.36 and B.37.



Key

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

[a] Enclosing rectangle. Bargraph cannot exceed boundaries of this rectangle.
[b] In this example, bargraph deflection is clockwise.

Figure B.9 — Arched bar graph object — Example

a) **Relationships:**

— **may contain** none (atomic);

— **may reference** macro, variable;

— **may be contained by** data mask, alarm mask, container.

b) **Allowed commands:**

— Change Numeric Value;

— Change Attribute;

— Change Size.

Table B.36 — Output arched bar graph events

Event	Caused by	VT behaviour	Message
On Refresh	See Data Mask Refresh for caused by conditions	Redraw this object.	—
On Change Value	Change Numeric Value command	Redraw this object, refresh parent mask.	Change Numeric Value Response
On Change Attribute	Change Attribute command	Redraw this object, refresh parent mask.	Change Attribute Response
On Change Size	Change Size command	Draw object at current location in background colour to erase it. Refresh parent mask.	Change Size Response

Table B.37 — Output arched bar graph attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=19	3	Object Type = Output Arced Bar Graph
Width	1	Integer	2	0-65535	4-5	Maximum width of the enclosing rectangle in pixels. Bar graph cannot exceed the bounds of this imaginary rectangle.
Height	2	Integer	2	0-65535	6-7	Maximum height of the enclosing rectangle in pixels. Bar graph cannot exceed the bounds of this imaginary rectangle.
Colour	3	Integer	1	0-255	8	Bar graph fill and border colour.
Target line colour	4	Integer	1	0-255	9	Target line colour (if drawn).
Options	5	Bitmask	1	0-31	10	Logical bits to indicate which parts to draw. 1 = True. Bit 0 = Draw border 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

Table B.37 (*continued*)

Attribute name	**AID**	**Type**	**Size bytes**	**Range or value**	**Record byte**	**Description**
Start angle	6	Integer	1	0-180	11	Start angle/2 (in degrees) from positive X axis anticlockwise (90° is straight up). Start and end angles define the arc. If the start and end angles are the same, the bar graph's arc is closed (360°).
End angle	7	Integer	1	0-180	12	End angle/2 (in degrees) from positive X axis anticlockwise (90° is straight up). Start and end angles define the arc. If the start and end angles are the same, the bar graph's arc is closed (360°).
Bar graph width	8	Integer	2	0-65535	13-14	Bar graph width in pixels. Bar graph width is with respect to the horizontal X axis. (see Figure B.9). Bar graph width shall be less than half the total width.
Min value	9	Integer	2	0-65535	15-16	Minimum value.
Max value	10	Integer	2	0-65535	17-18	Maximum value.
Variable reference	11	Integer	2	0-65534, 65535	19-20	Object ID of a number variable object in which to retrieve the bar graph's value. If this attribute is set to NULL (65535), the value is retrieved directly from the value attribute instead. Number variable's value shall be in the range 0-65535.
Value		Integer	2	0-65535	21-22	Current value. Used only if variable Reference attribute is NULL (65535). Bar graph fills or moves, depending on bar graph type, to a point calculated from this value and min/max values. If value > Max value or value < Min value, the bar graph is filled or shown empty and no error is generated by the VT.
Target value variable reference	12	Integer	2	0-65534, 65535	23-24	Object ID of a number variable object in which to retrieve the bar graph's target value. If this attribute is set to NULL (65535), the target value is retrieved directly from the Target value attribute instead. Number variable's value shall be in the range 0-65535.
Target value	13	Integer	2	0-65535	25-26	Current target value. Used only if target value variable Reference attribute is NULL (65535). Target value is displayed as a line on the bar graph to indicate some target or warning level. If Target value > Max value or Target value < Min value, the target line is shown on one of the ends of the bar graph and no error is generated by the VT.
Number of macros to follow		Integer	1	0-255	27	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Event ID}		Integer	1	0-255	28…	Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	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.38 and B.39.

a) **Relationships:**

— **may contain** none (atomic);

— **may reference** macro;

— **may be contained by** data mask, alarm mask, key, button, container, input list, auxiliary function, auxiliary input.

b) **Allowed command:**

— Change Attribute.

Table B.38 — 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.39 — Picture graphic attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=20	3	Object Type = Picture Graphic
Width	1	Integer	2	0-65535	4-5	Target width in pixels of the picture graphic. The height of the picture graphic is calculated from the Actual width/Height and this attribute to keep the same aspect and avoid distortion.
Actual width		Integer	2	0-65535	6-7	Actual width in pixels of the picture graphic raw data. VT shall scale the graphic to the size given by the width attribute.
Actual height		Integer	2	0-65535	8-9	Actual Height in pixels of the picture graphic raw data. VT shall scale the graphic to the size given by the width attribute.

Table B.39 (*continued*)

Attribute name	**AID**	**Type**	**Size bytes**	**Range or value**	**Record byte**	**Description**
Format		Integer	1	0-2	10	Picture graphic type: 0 = Monochrome; 8 pixels per byte. Each bit represents a colour palette index of 0 or 1. (“White” colour may vary with display hardware) 1 = 4 bit colour; 2 colour pixels per byte. Each nibble (4 bits) represents a colour palette index of 0 through 15. 2 = 8 bit colour; 1 colour pixel per byte. Each byte represents a colour palette index of 0 through 255. See Table A.3 for colour codes.
Options	2	Bitmask	1	0-7	11	Bit 0: 0 = Opaque, 1 = Transparent. If opaque, all pixels are drawn in indicated colour. Background objects do not show through. If transparent, pixels in the bitmap that have the transparency colour should show the colour of the background or objects underneath this picture graphic instead. Bit 1: 0 = Normal, 1 = Flashing. Flash style and rate determined by VT design. Bit 2: 0 = Raw data, 1 = Run-Length Encoded data (see B.12.2). This bit cannot be changed during runtime by Change Attribute command. (Any change will be ignored by the VT.)
Transparency colour	3	Integer	1	0-255	12	Pixels in the bitmap that have this colour index are transparent (background shows through).
# of bytes in raw data		Integer	4	0-2^32	13-16	Number of bytes in the raw data.
Number of macros to follow		Integer	1	0-255	17	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {raw data}		Integer	1	0-255	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 below. 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.
Repeat: {Event ID}		Integer	1	0-255	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	0-255	Depends on size of bitmap data	Macro ID of the macro to execute.

B.12.2 Picture graphic 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 6 bits in the second byte of each line would be filled with zero and the raw data would be 0 × FF, 0 × C0, 0 × FF, 0 × C0. 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.

In order to reduce the amount of data transmitted, a run-length encoding scheme can be used to compress the picture graphic data in realtime. 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, raw data should be transmitted instead and Bit 2 of the options attribute should 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 %.

The run-length algorithm is chosen because picture graphic data can be easily compressed and uncompressed in real time without the need for a buffer in the VT.

B.13 Variable objects

B.13.1 General

Variables are used to store a value that can be referenced and used by other objects. There are two types of variable object: number and string. Variables are referenced only, never directly included in a mask, key or container. See Table B.40.

a) **Relationships:**

— **may contain** none (atomic);

— **may reference** none;

— **may be contained by** none.

b) **Allowed commands:**

— Change Numeric Value;

— Change String Value (string variable object only).

Table B.40 — Variable events

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

B.13.2 Number variable

A number variable holds a 32-bit unsigned integer value. See Table B.41.

Table B.41 — Number variable attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=21	3	Object Type = Number Variable
Value		Integer	4	0-2^32	4-7	32-bit unsigned integer value.

B.13.3 String variable

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.42.

Table B.42 — String variable attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=22	3	Object Type = string variable
Length		Integer	2	0-65535	4-5	Maximum fixed length of the string value in characters.
Value		String			6…	String of characters. Pad with spaces as necessary to satisfy length attribute.

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.43 and B.44.

B.14.2 Font attributes

This object holds attributes related to fonts.

a) **Relationships:**

— **may contain** none (atomic);

— **may reference** macro;

— **may be contained by** none.

b) **Allowed commands:**

— Change Font Attributes;

— Change Attribute.

Table B.43 — 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.44 — Font attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=23	3	Object Type = Font Attributes
Font colour	1	Integer	1	0-255	4	Text colour.
Font size	2	Integer	1	0-14	5	Font size 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
Font type	3	Integer	1	0,1 or 255	6	0 = ISO Latin 1 1 = ISO Latin 9 255 = Proprietary

Table B.44 (*continued*)

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Font style	4	Bitmask	1	0-127	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 [a] Bit 5 = 1 = Flashing between Inverted and styles set by bits 0-3 Bit 6 = 1 = Flashing between Hidden and styles set by bits 0-3. Bit 6 has priority over bit 5. Bit 7 = 1 = Not Used
Number of macros to follow		Integer	1	0-255	8	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Event ID}		Integer	1	0-255	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	0-255	10…	Macro ID of the macro to execute.

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

B.14.3 Line attributes

This object holds attributes related to output shape objects. See Tables B.45 and B.46 and Figure B.10.

a) **Relationships:**

- **may contain** none (atomic);
- **may reference** macro;
- **may be contained by** none.

b) **Allowed commands:**

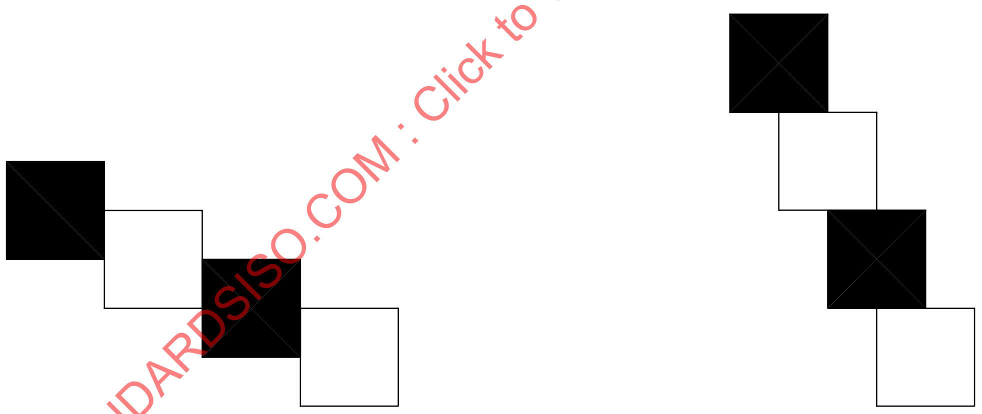
- Change Line Attributes;
- Change Attribute.

Table B.45 — Line attributes events

Event	Caused by	VT behaviour	Message
On Change Line Attributes	Change Line Attributes command	Redraw all objects that are currently displayed and reference this object. Refresh parent object.	Change Line Attributes Response
On Change Attribute	Change Attribute command	Redraw all objects that are currently displayed and reference this object. Refresh parent object.	Change Attribute Response

Table B.46 — Line attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=24	3	Object Type = Line Attributes
Line colour	1	Integer	1	0-255	4	Pen colour.
Line width	2	Integer	1	0-255	5	Pen thickness in pixels. Lines are drawn with a square paintbrush of this size.
Line art	3	Bitmask	2	0-65535	6-7	Bit pattern art for line. Each bit represents a paintbrush spot. Zero (0) bits are skipped (background colour) and one (1) bits are drawn in the line colour. Each bit is the size of the current paintbrush. For example, 00110011 would represent two skipped paintbrush spots followed by two paintbrush spots drawn and so on. See Figure B.10
Number of macros to follow		Integer	1	0-255	8	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Event ID}		Integer	1	0-255	9…	Event ID of event type that causes this macro to execute.
{Macro ID}		Integer	1	0-255	10…	Macro ID of the macro to execute.



a) More horizontal line

b) More vertical line

Figure B.10 — Effect of line art attribute — Example pattern: 1010…

B.14.4 Fill attributes

This object holds attributes related to filling output shape objects. See Tables B.47 and B.48.

a) **Relationships:**

— **may contain** none (atomic);

— **may reference** macro;

— **may be contained by** none.

b) **Allowed commands:**

— Change Fill Attributes;

— Change Attribute.

Table B.47 — 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.48 — Fill attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=25	3	Object Type = Fill Attributes
Fill type	1	Integer	1	0-3	4	0 = no fill 1 = fill with line colour 2 = fill with specified colour in fill colour attribute 3 = fill with pattern given by fill pattern attribute
Fill colour	2	Integer	1	0-255	5	Colour for fill if type = 2.
Fill pattern	3	Integer	2	0-65534, 65535	6-7	Object id of a picture graphic object to use as a fill pattern if type = 3. Otherwise NULL (65535). **IMPORTANT — In order to simplify 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).**

Table B.48 (*continued*)

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Number of macros to follow		Integer	1	0-255	8	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Event ID}		Integer	1	0-255	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	0-255	10…	Macro ID of the macro to execute.

B.14.5 Input attributes

This object defines the valid or invalid characters for an input string object. The VT shall check this object for valid characters and not permit operator entry of invalid characters into the input field. This object is referenced by an input string object. See Tables B.49 and B.50.

a) **Relationships:**

— **may contain** none (atomic);

— **may reference** macro;

— **may be contained by** none.

b) **Allowed command:**

— Change String Value.

Table B.49 — Input attributes events

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

Table B.50 — Input attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=26	3	Object Type = Input Attributes
Validation type		Integer	1	0-1	4	0 = valid characters are listed 1 = invalid characters are listed
Length		Integer	1	0-255	5	Length of validation string.
Validation string		String			6…	String containing all valid or invalid character codes (depends on validation type attribute).
Number of macros to follow		Integer	1	0-255	Depends on size of string	Number of macro references included even if zero. Each macro reference consists of 2 bytes: one for event ID and one for macro ID. Whenever the indicated event occurs, the associated macro is executed.
Repeat: {Event ID}		Integer	1	0-255	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	0-255	Depends on size of string	Macro ID of the macro to execute.

B.15 Object pointer

The object pointer object allows run-time modification of included objects. By changing the value of the pointer object, a different object can be referenced to the same location. The type of object that the pointer object is allowed to point to is limited and depends on the parent object. Refer to the mask, key, button and container descriptions for a list of the objects that can be pointed to. An object pointer can point to the NULL object ID and in this case nothing would be drawn. See Tables B.51 and B.52.

a) **Relationships:**

— **may contain** none (atomic);

— **may reference** none;

— **may be contained by** data mask, alarm mask, soft key mask, key, button, container.

b) **Allowed command:**

— Change Numeric Value.

Table B.51 — 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.52 — Object pointer attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-65534, 65535	1-2	Object identifier. Shall be unique within the object pool.
Type		Integer	1	=27	3	Object Type = pointer
Value		Integer	2	0-65534, 65535	4-5	Object ID of a referenced object or $FFFF_{16}$ for NULL object ID.

B.16 Macro object

Macros are used to define a list of commands that can be referenced by an event. A macro is defined by a series of one or more command packets. Macro object IDs shall be 255 or less.

NOTE Command packets are defined in Annex F but not all commands are allowed in a macro.

See Table B.53.

a) **Relationships:**

— **may contain** none;

— **may reference** none;

— **may be contained by** none.

b) **Allowed command:**

— none.

Table B.53 — Attributes and record format

Attribute name	AID	Type	Size bytes	Range or value	Record byte	Description
Object ID		Integer	2	0-255	1-2	Object identifier. 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		Integer	1	=28	3	Object Type = Macro
# of bytes to follow		Integer	2	0-65535	4-5	Number of bytes to follow. All commands are 8 bytes in length except for the change string value command, which is variable.
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.

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 may be via PC Cards, or diagnostic or programming tools.

A working set's object pool must 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 object pool exceeds the 1 785 byte limit of normal transport protocol, the extended functions of transport protocol shall be used. The VT design shall be able to support all the transport protocol functions.

The VT shall receive, parse and store the received objects. 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 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 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 command (see Annex D). The VT acknowledges this message with a get memory response (see Annex D). 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 described in C.2.3. Normal handshaking, error checking and re-transmission, in accordance with ISO 11783-3, shall be implemented. Working set designers should recognize that the VT can send CTS with number of packets set to zero (0) while other object pools are being loaded and that this could continue for a significant amount of time.

   The following rules govern the transfer of the object pool:

   1) The working set master 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 must 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.

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 object pool is now complete and ready for use.

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 (refer to 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 message

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.

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 4-7 = 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 command (see D.2). 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 transport protocol to move the object or objects to the VT. Normal handshaking, error checking and retransmission, in accordance with ISO 11783-3, shall be implemented. The rules given in C.2.2 b) apply.

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.

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.

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.

D.2 Get Memory

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 Size
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}	

D.3 Get Memory Response

If the VT responds with status code one (1), the working set master is not permitted to 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 Size Response
Byte 2	Version number		The version 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	The version of the first edition published as an International Standard, and so on	
Byte 3	Status	0	There may be enough memory. However, because there is overhead associated with object storage it is impossible to predict whether there is enough memory available.	
		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

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
Bytes 2-4			Reserved, transmit as FF16	
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	
Byte 8	Physical soft keys		Number of physical soft keys	

D.6 Get Text Font Data

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:	In response to Get Text Font Data message
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			
		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 between hidden and styles set by bits 0-3	
		1000 0000	Not used	

D.8 Get Hardware

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			Reserved, transmit as FF_{16}	

Byte	3	Graphic type	Supported graphic modes 0 = Monochrome (VT supports colour codes 0 and 1 and monochrome picture graphic objects only) 1 = 16 Colour (VT supports colour codes 0 through 15 and monochrome and 16 colour picture graphic objects). 2 = 256 Colour (VT supports colour codes 0 through 255 and all formats of picture graphic objects).
Byte	4	Hardware	Bit 0 = 1 = VT has a touch screen Bit 1 = 1 = VT has a pointing device Bit 2 = 1 = VT has multiple frequency audio output Bit 3 = 1 = VT has adjustable volume audio output
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

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 message to get its object pool copied from non-volatile storage into volatile storage. The availability and organization of the non-volatile storage area is VT-specific. Storing and restoring an object pool includes all object definitions. There shall be a method inside the VT to assign a stored object pool uniquely to a specific working set. Dependant 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 VT, each pool should be identified by a seven-character-wide version label.

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 following messages are not allowed in macros.

E.2 Get Versions

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	

E.4 Store Version

The store version message 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	

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 message
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

The Load Version message allows a working set to load a copy of a 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.

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	

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 message
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			
	Bits 7-4	0000	Not used	
	Bits 3-0	0000	Successfully loaded	
		0001	Not used	
		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

The Delete Version message 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.

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	

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 message
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}	

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. The working set shall wait for a response up to a maximum of 1,5 s before sending another command. Each of the commands in the present 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).

F.2 Hide/Show Object

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
Byte	2,3			Object ID	
Byte	4			0 = Hide, 1 = Show	
Byte	5-8			Reserved, transmit as FF_{16}	

F.3 Hide/Show Object Response

The VT uses this message to respond to a hide or show event.

Transmission repetition rate:	In response to Hide/Show Object message
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
Byte	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

This command is used to enable or disable an input field object and pertains to the accessibility of an input object.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a macro:	Yes

Byte 1	VT function = 161_{10}				
	Bits 7-4	1010	Command	Command
	Bits 3-0	0001	Parameter	Enable/Disable object
Byte 2,3			Object ID	
Byte 4			0 = Disable, 1 = Enable	
Byte 5-8			Reserved, transmit as FF_{16}	

F.5 Enable/Disable Object Response

The VT uses this message to respond to an enable or disable event.

Transmission repetition rate:	In response to an Enable/Disable Object message
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
Byte 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

This command is used to force the selection of an input field object. The VT shall provide a way for the operator to recognize a selected input field object. If the input object is disabled or not visible, an error code is returned.

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
Byte 2,3			Object ID	
Byte 4-8			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 message
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			0 = object is Deselected, 1 = object is Selected	
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 visible Bit 3 = 1 = Could not complete. Another Input field is currently being modified. Bit 4 = 1 = Any other error	
Bytes 6-8			Reserved, transmit as FF_{16}	

F.8 ESC

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
Byte 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 message
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
Byte 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, ESC ignored. Bits 1-3 = Not used Bit 4 = 1 = Any other error	
Bytes 5-8			Reserved, transmit as FF_{16}	

F.10 Control Audio Device

This command can be used to control the audio on the VT.

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		Number of repetitions	
Byte 3,4		Frequency in hz	
Byte 5,6		On-time duration in ms	
Byte 7,8		Off-time duration in ms	

F.11 Control Audio Device Response

This message is sent by the VT in response to a control audio command.

Transmission repetition rate:	In response to Control Audio Device message
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	
Byte 3-8		Reserved, transmit as FF_{16}	

F.12 Set Audio Volume

This command can be used to control the audio on the VT.

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		% (0-100 %) of maximum volume	
Byte 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 message
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	
Byte 3-8			Reserved, transmit as FF_{16}	

F.14 Change Child Location

The Change Child Location command is used to change the position of an object. Since the object may be included in many parent objects, the parent object ID is also included. This command is for small position changes for producing simple animation. When the object is moved, the parent object shall be refreshed. The position attributes given in the message provide a relative change in position and 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 of top left corner of object	
Byte 7			Relative change in Y position of top left corner of object	
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 message
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	
			Bit 2 = 1 = Invalid X Position	
			Bit 3 = 1 = Invalid Y Position	
			Bit 4 = 1 = Any other error	
Bytes 7,8			Reserved, transmit as FF_{16}	

F.16 Change Child Position (transport protocol)

The Change Child Position command is used for large changes of the position of an object. This command is used to change the position of the child object by larger values than in Change Child Location. Since the object may be included in many parent objects, the parent Object ID is also included. When the object is moved, the parent object shall be refreshed. The position attributes given in the message are signed integer. Positive values indicate a position 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:	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			Relative X position of top left corner of object (relative to the top left corner of the container object)	
Bytes 8,9			Relative Y position of top left corner of object (relative to the top left corner of the container object)	

F.17 Change Child Position Response

This message is sent by the VT in response to the Change Child Response message.

Transmission repetition rate:	In response to Change Child Position message
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			Bit 0 = 1 = Invalid Parent Object ID	
			Bit 1 = 1 = Invalid Object ID	
			Bit 2 = 1 = Invalid X Position	
			Bit 3 = 1 = Invalid Y Position	
			Bit 4 = 1 = Any other error	
Bytes 7,8			Reserved, transmit as FF_{16}	

F.18 Change Size

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 message
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 = 1 = Invalid Width Bit 2 = 1 = Invalid Height Bit 3 = not used Bit 4 = 1 = Any other error	
Bytes 5-8			Reserved, transmit as FF_{16}	

F.20 Change Background Colour

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

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a macro:	Yes

Byte 1	VT function = 167_{10}			
	Bits 7-4	1010	Command	Command
	Bits 3-0	0111	Parameter	Change Background Colour
Bytes 2,3			Object ID of object to change	
Byte 4			New Background colour (see Table 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 message
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 Table 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

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.

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 field:	1 byte for true/false
			Number input field:	4 bytes for integer input
			List input field:	1 byte for list index
			Number output field:	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 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 command.

Transmission repetition rate:	In response to Change Numeric Value message
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 2-3 = 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 field:	1 byte for true/false
			Number input field:	4 bytes for integer input
			List input field:	1 byte for list index
			Number output field:	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

F.24 Change String Value (transport protocol)

String attributes shall be handled with the transport protocol (see ISO 11783-3), since the length of the string could exceed the length of a CAN packet. 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.

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	0100	Parameter	Change String Value
Bytes 2,3			Parent Object ID	
Byte 4,5			Object ID of object to move	
Byte 6			Bit 0 = 1 = Invalid Parent Object ID	
			Bit 1 = 1 = Invalid Object ID	
			Bit 2 = 1 = String too long	
			Bit 3 = 1 = Any other error	
Bytes 7,8			Reserved, transmit as FF_{16}	

F.26 Change End Point

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 message
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

This command is used to change the font attributes in a font attributes object.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a macro:	Yes

Byte 1	VT function = 170_{10}		
	Bits 7-4 1010	Command	Command
	Bits 3-0 1010	Parameter	Change Font Attributes
Bytes 2,3		Object ID of object to change	
Byte 4		Font Colour (see Table 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 message
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

This command is used to change the line attributes in a line attributes object.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a macro:	Yes

Byte 1	VT function = 171_{10}		
	Bits 7-4 1010	Command	Command
	Bits 3-0 1011	Parameter	Change Line Attributes
Bytes 2,3		Object ID of object to change	
Byte 4		Line Colour (see Table 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 Change Line Attributes command.

Transmission repetition rate:	In response to the Change Line Attributes message
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

This command is used to change the fill attributes in a fill attributes object.

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	ECU to VT, Destination-Specific
Allowed in a macro:	Yes

Byte 1	VT function = 172_{10}			
	Bits 7-4	1010	Command	Command
	Bits 3-0	1100	Parameter	Change Fill Attributes
Bytes 2,3			Object ID of object to change	
Byte 4			Fill Type	
Byte 5			Fill Colour (See Table 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 message
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

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 also the VT Change Active Mask message in Annex H.

Transmission repetition rate:	In response to Change Active Mask message
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		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 = Missing Objects Bit 3 = 1 = Mask or child object has errors Bit 4 = 1 = Any other error	
Bytes 5-8		Reserved, transmit as FF_{16}	

F.36 Change Soft Key Mask

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	Change Soft Key Mask
Byte 2		Mask Type (1=Data, 2 = Alarm)	
Bytes 3,4		Data or Alarm Mask Object ID	
Byte 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 command. See also the VT Change Soft Key Mask message in Annex H.

Transmission repetition rate:	In response to Change Soft Key Mask message
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

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