MOST

Media **O**riented **S**ystems **T**ransport

Multimedia and Control Networking Technology

MOST FunctionBlock GraphicDisplay
Rev 2.3
09/2003

Legal Notice

COPYRIGHT

LICENSE DISCLAIMER

Nothing on any MOST Cooperation Web Site, or in any MOST Cooperation document, shall be construed as conferring any license under any of the MOST Cooperation or its members or any third party's intellectual property rights, whether by estoppel, implication, or otherwise.

CONTENT AND LIABILITY DISCLAIMER

MOST Cooperation or its members shall not be responsible for any errors or omissions contained at any MOST Cooperation Web Site, or in any MOST Cooperation document, and reserves the right to make changes without notice. Accordingly, all MOST Cooperation and third party information is provided "AS IS". In addition, MOST Cooperation or its members are not responsible for the content of any other Web Site linked to any MOST Cooperation Web Site. Links are provided as Internet navigation tools only.

MOST COOPERATION AND ITS MEMBERS DISCLAIM ALL WARRANTIES WITH REGARD TO THE INFORMATION (INCLUDING ANY SOFTWARE) PROVIDED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

In no event shall MOST Cooperation or its members be liable for any damages whatsoever, and in particular MOST Cooperation or its members shall not be liable for special, indirect, consequential, or incidental damages, or damages for lost profits, loss of revenue, or loss of use, arising out of or related to any MOST Cooperation Web Site, any MOST Cooperation document, or the information contained in it, whether such damages arise in contract, negligence, tort, under statute, in equity, at law or otherwise.

FEEDBACK INFORMATION

Any information provided to MOST Cooperation in connection with any MOST Cooperation Web Site, or any MOST Cooperation document, shall be provided by the submitter and received by MOST Cooperation on a non-confidential basis. MOST Cooperation shall be free to use such information on an unrestricted basis.

TRADEMARKS

MOST Cooperation and its members prohibit the unauthorized use of any of their trademarks. MOST Cooperation specifically prohibits the use of the MOST Cooperation LOGO unless the use is approved by the Steering Committee of MOST Cooperation.

SUPPORT AND FURTHER INFORMATION

For more information on the MOST technology, please contact:

MOST Cooperation
Administration
Bannwaldallee 48
D-76185 Karlsruhe
Germany

Tel: (+49) (0) 721 966 50 00
Fax:(+49) (0) 721 966 50 01
E-mail: contact@mostcooperation.com
Web: www.mostcooperation.com

Bibliography MOST Function Catalog

This is a list of released FunctionBlocks at the release time of this specification. FBlocks which are released later are not reflected in this list.

FBlockID	FunctionBlock
0x00	GeneralFBlock
0x00	GeneralPlayer
0x01	NetBlock
0x02	NetworkMaster
0x03	ConnectionMaster
0x06	Diagnosis
0x0F	Enhanced Testability
0x22	AudioAmplifier
0x26	MicrophoneInput
0x30	AudioTapePlayer
0x31	AudioDiskPlayer
0x34	DVDVideoPlayer
0x40	AmFmTuner
0x41	TMCTuner
0x42	TVTuner
0x50	Telephone
0x51	GeneralPhoneBook
0x60	GraphicDisplay
0xFF	UniqueFunctions

GraphicDisplay FBlock (0x60) Change History

Changes GraphicDisplay FBlock 2.3 to GraphicDisplay FBlock x.x.x

Change Ref.	FktID	Changes
-	-	-
		-
		-

1 Introduction

A MOST Function Catalog is a collection of MOST FunctionBlocks.

This document contains the specification of a FunctionBlock. MOST FunctionBlocks are standardized and maintained by MOST workgroup Device Architecture (WG_DA). In order to speed up the process of making new Function Blocks available, every Function Block will be updated individually as required.

2 FunctionBlock Specification

2.1 Design Overview

One display is represented by one function block, that is, one instance of the FBlock Graphic Display (FBlockID = 0x12). Through the *control interface* of this FBlock, it is possible to create a view port on the display. Associated with each view port is exactly one *socket connection*, implemented through the MOST High Protocol running on the asynchronous channel of the MOST ring.

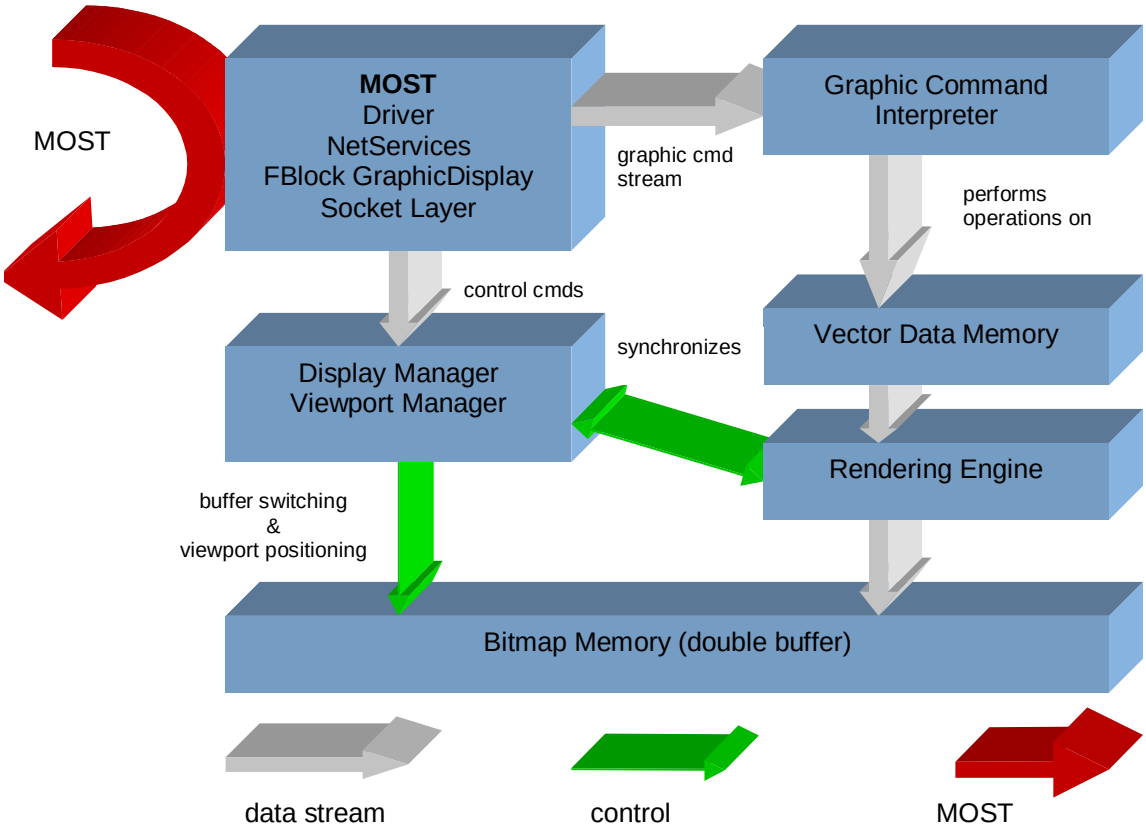

Figure 2-1: ***Basic Model of the Graphics Engine***

Other socket-oriented protocols can be possible and are considered in the parameters of the command that creates view port and socket. The socket connection provides the transport channel for

the actual graphical data to the view port. The source of the data is the *application*, which has not to be represented by a special MOST FBlock for the purposes of the graphics interface.

The graphical data consists of command sequences for drawing two-dimensional vector *elements* like lines, polygons, etc. as well as command sequences for manipulating the previously drawn graphics. Single elements cannot be manipulated directly; instead, they have to be drawn into a *container*, which can then be manipulated (moved, rotated, scrolled, etc.). Containers serve three basic purposes:

1. Grouping. Elements are grouped through their membership in one container.
2. Layering. Containers have a defined Z-order relationship to each other.
3. Clipping. Each container has a rectangular clipping region associated with it.

Containers can also be elements of other containers. This hierarchical structure can be used through many levels of recursion, if desired. If in any specific implementation the maximum depth of the hierarchy is reached, an appropriate error message is returned. This maximum number can be queried in advance.

Even though it is not part of this specification, the graphics engine implemented in the display device is supposed to be modeled like the one in figure 1.

Drawing operations can take place anywhere in the valid coordinate space (±32767 × ±32767). Still, drawn elements are only visible to the user if they are not clipped, of course. By scrolling the contents of a container, previously invisible elements can move into the visible area.

2.2 Protocol Structures

2.2.1 Control Commands

The protocol structures used for control commands are the same as for every MOST Function Block (see MOST specification). The functions for the FBlock GraphicDisplay can be found in section 2.8.1 on page 19.

2.2.2 Data Commands

Since the graphic commands are transported through a transparent data channel, it is not necessary to design them in compliance to the standard MOST command architecture; nevertheless, the transparent data stream itself is encapsulated in standard MOST protocol telegrams (Property GraphicData uses MOST High Protocol). The protocol structure for the graphic commands is designed as follows:

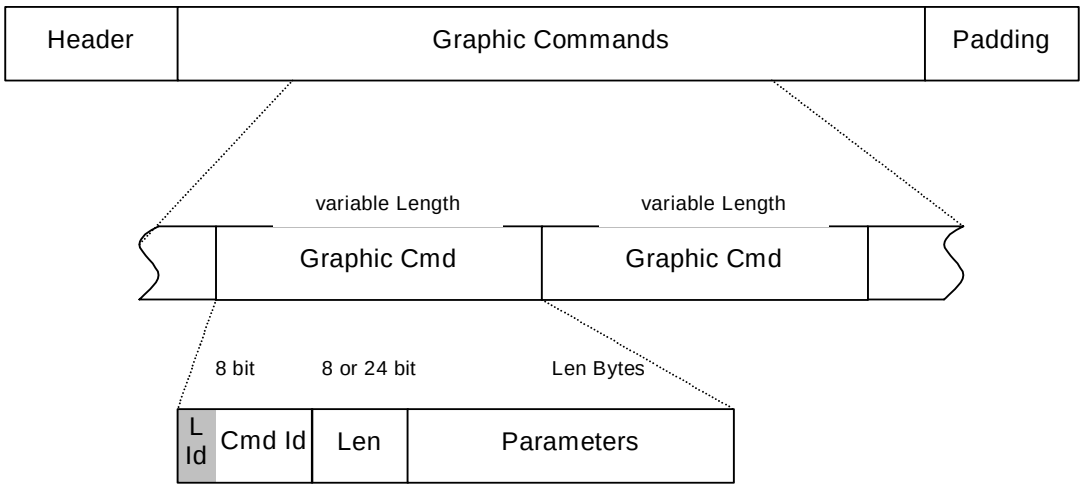

*Figure 2-2**: Data Channel Protocol Structure***

1. Header: The header of the surrounding transport protocol (MOST High Protocol). One command may be divided into more than one packet of the transport protocol as long as this protocol guarantees the correct re-assembly. For MOST High Protocol transmission, Header refers to the complete MOST High Protocol header plus the Handle parameter (v. section 2.8.12: GraphicData).

2. Padding: No-Op Commands (0x00) to pad remaining bytes in a telegram if the transport telegram requires a certain length. For MOST High Protocol transmission, no padding is needed.

3. Cmd Id: Identifies the command. For a list of commands, refer to section **Fel! Hittar inte referenskälla.**: **Fel! Hittar inte referenskälla.**. Note that the most significant bit of the Cmd Id is also the Length indicator L Id; if it is set to 1, the Len field is 24 bits wide, instead of 8 bits. Thus, commands with an Id less than 128 can have 255 parameter bytes in maximum, whereas a command with an Id greater than 127 can have up to 16 Megabytes of parameter data (This theoretical maximum will be subject to an additional limitation through the standardization process). Since the L Id flag is part of the Cmd Id, two commands with identical Id except for the L Id flag are regarded as different commands. In some cases, there are several versions of the same command, depending on the set of given parameters. The Id ranges 0x70 – 0x7F and 0xF0 – 0xFF are reserved for proprietary extensions.

4. Len: The length of the parameter field in bytes. Can be zero. There are two Commands that have no Len field: Cmd Id = 0x00 and Cmd Id = 0x80.

5. Parameter: Parameters for the respective command, as described in the reference part. This field is not present for Len = 0 or if there is no Len field.

2.3 Viewports and Sockets

The FBlock GraphicDisplay provides control commands for creating and deleting Viewports. There is no complete set of functionality for managing Viewports like in a window manager, since the philosophy behind this reaches far into the “look and feel” of the graphical user interface and hence is not part of this standardization. A Viewport is identified by its handle, a unique ID that is returned by the command that creates the Viewport.

2.4 Coordinate System and Color Palettes

2.4.1 Coordinate System

The physical resolution of the display can be retrieved by reading out the property PhysicalMetrics. It contains the X and Y resolution of the display, as well as its width and height in millimeters and a bit field indicating the available color/palette modes. After a view port has been created with the given width and height, the coordinate system of its root container (see below) is set to match the physical pixel resolution of the display, with the origin in the upper left corner of the view port and positive values going to the right and downwards. Each container holds a local coordinate system for its elements that is defined on creation of the container and consists of an origin, an orientation and two scaling factors. Only the origin can be specified on container creation; default values are taken for the other parameters and can be adjusted afterwards by the commands RotateContainer and SetScale.

2.4.2 Color Palettes

A bit field indicates the color modes of which the display is capable. The actual color mode is specified through the ColorMode property. Depending on whether a palette-based or a true-color-based mode has been chosen, the commands for palette management are valid or not. Palettes are global for all Viewports and can be created, destroyed and modified through control channel methods and properties. Associated with each palette is a string that can be used to give the palette a unique name (Some standard palettes and their names should be standardized within MOST). Colors are always specified in 32bit (RGBα) format and reduced automatically by the display if a mode with less bit depth is used. The background color of each view port is set through the BgColor command.

2.5 Containers

2.5.1 Purpose

As mentioned in section 2.1, containers serve the three basic purposes grouping, layering and clipping. Containers can contain graphical elements like lines, circles, etc. and sub-containers. When manipulating a container, all elements – graphical elements, sub-containers and their elements – are affected. The contents of a container can be copied into another one – hence hidden containers could serve as image or icon well, for example (This replaces the class or macro / template mechanism proposed by older drafts).

2.5.2 Hierarchical ID Structure

Within one view port one container is the *current* or *selected* container. It can be changed with the SelectContainer command (v. section **Fel! Hittar inte referenskälla.**). The subsequent commands are then related to that container. After creation of the view port, there exists one predefined container; the *root container*.

The nesting of containers within other containers results in a hierarchical tree structure. Containers are identified by a unique ID that derives from its position in the tree.

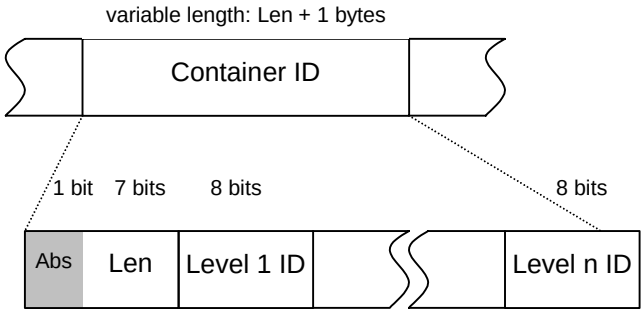

*Figure 2-3**: ID Field Structure***

Figure 2-3 shows the structure of the container ID. The first byte contains information about how many of the subsequent bytes belong to this ID and whether it is a relative or absolute ID. Absolute IDs always describe a complete path through the tree (Abs=1), whereas a relative path describes the path from the current container to the specified one (Abs=0). Each level of the hierarchy can hold up to 254 containers, since the values 0 and 0xFF are reserved. 0xFF is used in the relative ID to specify the parent container. Therefore, the root container is represented by an ID with just the Abs/Len byte, containing 0x80. It is also possible to specify the current container, which is also just the Abs/Len byte, containing 0x00. A complete ID is called FULLID in the reference part. There are also some commands that take a partial ID as a parameter; this is called IDPART in the reference part. A partial ID is understood as a relative path from the current container to one of its direct children and hence is always one byte long. An exception is the SetZOrder command (v. section **Fel! Hittar inte referenskälla.**), where an IDPART argument is used to specify a sibling rather than a child.

2.5.3 Creation

A container is created as an element of another container through the CreateContainer command. The first container created by the application, if any, is always an element of the root container. There are two versions of the CreateContainer command: The first one creates a new container as a child (or *subcontainer*) of the current container, the second one takes a fully qualified ID to create a container anywhere in the tree. After creation, the new container is automatically selected.
When creating a container, the origin of the new container's coordinate system, given as a point in the parent container's coordinate system, has to be specified, along with its initial flag settings. For a description of the flags, refer to section **Fel! Hittar inte referenskälla.**.

2.5.4 Clipping

Each container has a rectangular clipping region associated with it. By default, this region is as large as the complete coordinate space, that is $\pm 32767 \times \pm 32767$. An alternative clipping region can be set with the SetClipRegion command. Clipping can be useful e.g. for creating window-like screen setups within the view port or to prepare graphics in a container outside the visible area that can be scrolled into view.

2.5.5 Moving, Rotating and Scaling

By moving a container with the MoveContainer command, everything that is related to the container (origin, clipping region, all elements) is moved by the specified amount in the coordinate system of the parent container.
By rotating a container with the RotateContainer command, everything that is related to the container (origin, clipping region, all elements) is rotated by the specified amount of degrees around the specified center point in the coordinate system of the parent container. If the KeepOrientation flag is set, the elements retain their orientation on rotation (e.g. remaining upright) and textures are not rotated. The angle for the rotation is specified in an 8-bit signed value which can be calculated through the formula $angle_{argument} = angle_{degrees} \cdot 128 / 180°$, with the following prominent values:

Arg	Deg	Absolute	Relative
0	0	Up	No rotation
64	90°	Right	Clockwise
±128	180°	Down	Flip
-64	-90°	Left	Counterclw.

The same angle data type is used for the RotateContent, Arc and AngleArc commands. Scaling means manipulation of the container's X and Y scale factors, which affects either only the vectors of the elements (e.g. for zooming maps, where streets and labels should retain their width) or results in a true-to-scale enlargement or reduction, depending on the ZoomMode flag.

2.5.6 Content Scroll & Rotation

Content Scroll means a movement of each single element of the container, whereas the container's origin and clipping region remain unchanged. Content Rotation means a rotation of each single element of the container around a specified center point by a specified angle, whereas the container's orientation remains unchanged. The KeepOrientation flag applies for the angle specification, v. above.

2.5.7 Copying

The complete contents of a container can be copied into another one. Note that this is a real duplication, so every sub- and sub-sub-container, etc. is instantiated another time (Implementations are free to organize their internal memory structures in a smarter way, of course). The clipping region of the container is not copied, whereas those of the sub-containers are. The containers must be in different branches of the tree to avoid recursion.

2.5.8 Z-Order

Sibling containers have a Z-order relationship to each other. By default, the Z-order matches the order of creation, so the last created container is the topmost. This can be changed by applying the SetZOrder command to a container. It sets the container in front (or on top) of another specified one. If a container shall be behind all its siblings, the pseudo-value 0 is specified. If a container shall be on top of all others, the pseudo-value 0xFF is specified. Primitive elements are always behind all children of their container and have a constant Z-order relationship to each other, which is the order of creation. To get the Z-order of primitives changed, or to get them on top of children, they must be put in a subcontainer instead.

2.5.9 Flags

Each container owns an 8-bit field containing flags with individual meaning. Those flags include, but are not limited to, the KeepOrientation flag as well as the ZoomMode flag and will be explained in detail in the reference part. Text field containers carry a second 8-bit flag field for alignment control purposes.

2.6 Drawing Graphics

2.6.1 Pens and Brushes

To define the color and style in which graphic primitives are drawn, pens and brushes are used. A pen describes the color and style in which lines and outlines of shapes are drawn. A brush describes the color, pattern and / or bitmap texture that is used to fill a shape. Available styles are very similar to the ones provided by the Win32 API. There is always only one brush and one pen defined for a container. It is not possible to create additional pens or brushes; instead, the only one that exists for every respective container has to be redefined. The default settings for pens are: Color 0, width 1, solid line, square ends, miter join. The default settings for brushes are: Color 0, 100% fill. Images uploaded to the display (see section 2.6.3) can serve as textures.

2.6.2 Vector Graphics

Vector oriented drawing is the main purpose of this graphics API. Numerous vector commands are available, with varying number of arguments. Most of those commands resemble those provided by the Win32 API, just that the primitives are not simply drawn by the display (that is, converted to bitmap graphics) but are memorized by it and can be redrawn anytime. Since the single vector elements do not have any Id or handle assigned to them when they are created, it is not possible to manipulate them individually afterwards. Instead they would have to be put into a subcontainer.

2.6.3 Bitmap Graphics

Images can be uploaded into the display's memory in various formats. Every display should support at least one image file format of each of the following categories:

1. Photo-realistic images, e.g. JPEG images.
2. Graphics with loss-less compression, e.g. GIF or PNG images.
3. Simple, run-length-encoded bitmaps like Windows RLE or BMP files (Bosch offered to provide an advanced RLE file format called MG98, which will eventually be incorporated into an appendix of this proposal / specification).

The recommended image format set to be supported is: JPEG, PNG, MG98. Images are instantiated on the screen with the DrawImage command or can be used as source for texture mapping (see UseBrush).

2.7 Drawing Text

2.7.1 Textfield Element

A text field is a special form of element that has its own Id, just like containers. It provides some additional properties like font and alignment settings and, of course, a string buffer containing the text that has to be displayed. The text buffer must support all codings that are defined in the MOST specification.

2.7.2 Alignment

Two different methods of alignment are available. The first one acts like the commonly known text boxes in word processing applications: On creation of the text field, the corners of a box are specified, and text can be aligned in the box as usual (left-justified, right-justified, centered, block, same vertically). The second one aligns the string on a virtual Bezier line, hence providing the possibility of embedding a street name into the street on the map, for example. The coordinates of the Bezier points are given in the box's local coordinate system.

2.7.3 Rotation and Autoflip

There are two methods to keep the text from becoming unreadable upon rotation:

1. Use the KeepOrientation flag. It works just like the one of containers, so the text remains always upright or in whatever orientation it has been set before setting the flag to '1'. This method works especially for box-aligned texts.

2. Use the Autoflip flag. If set to '1', this flag instructs the display to automatically flip the text if it tends to become unreadable due to rotation. The exact criteria of readability is implementation specific.

2.7.4 Fonts

A display should provide a version of each of the following standard fonts:

1. Proportional, without serifs (e.g. Arial)
2. Proportional, with serifs (e.g. Times)
3. Monospaced, with serifs (e.g. Courier)
4. Monospaced, without serifs

These fonts should be scalable. Additional fonts can be implemented on a proprietary or bilateral base, or can be uploaded with the LoadFont command. Uploaded fonts are bitmap fonts and hence are not scalable.

2.7.5 Character Sets

The general MOST definition of character sets shall apply. This means that in the first byte of the string, an identifier of the used character set (like standard ASCII or ISO Latin-1 or a MBCS form) is given (The enumeration of supported character sets is specified in the MOST Specification document).

2.8 Reference

2.8.1 Control Channel Commands

Besides the mandatory functions of an FBlock (like FktIDs, Notification, etc.), the FBlock GraphicDisplay provides the following functions:

2.8.2 PhysicalMetrics

PhysicalMetrics contains a description of the display's hardware properties like resolution, size and color depth.

Function class: Record of { Number Number Number Number Boolean Boolean }

FBlock	FktID	OPType	Parameter
GraphicDisplay (0x60)	PhysicalMetrics (0x200)	GetInterface	
		Get	Pos
		Status	Pos, Data
		Interface	Flags, Class, Name, NMax, Class, OPTypes, Name, Units, DataType, Exponent, Min, Max, Step
		Error	ErrorCode, ErrorInfo

Parameter

Pos

The parameter Pos={x,y} consists of Bytes and indicates the parameters to be retrieved and/or modified. Since the construct has only one dimension, the second byte remains unused.

Basis DataType	Exp	Range	Step	Unit
Unsigned Word	0	full range	1	none

Data

The contents of Data depends on the setting of the Pos parameter.

Basis datatype	Length	Description	
Stream	-	Pos	Data
		{x=0, y=0}	{ Xres, YRes, XSize, YSize, ColorModes, Flags }

XRes

Horizontal display resolution in pixels.

Basis datatype	Exp	Range	Step	Unit
Unsigned Word	0	full range	1	none

YRes

Vertical display resolution in pixels.

Basis datatype	Exp	Range	Step	Unit
Unsigned Word	0	full range	1	none

XSize

Horizontal display size in millimeters.

Basis datatype	Exp	Range	Step	Unit
Unsigned Word	-3	full range	1	m

YSize

Vertical display size in millimeters.

Basis datatype	Exp	Range	Step	Unit
Unsigned Word	-3	full range	1	m

ColorModes

Available color modes.

Basis datatype	Bit-Nr.	Code	Description
Boolean	Bit 0	FALSE	TrueColor 32 bit unsupported
		TRUE	TrueColor 32 bit supported
	Bit 1	FALSE	TrueColor 24 bit unsupported
		TRUE	TrueColor 24 bit supported
	Bit 2	FALSE	HighColor 16 bit unsupported
		TRUE	HighColor 16 bit supported
	Bit 3	FALSE	8 bit palette / 32 bit color depth unsupported
		TRUE	8 bit palette / 32 bit color depth supported
	Bit 4	FALSE	8 bit palette / 24 bit color depth unsupported
		TRUE	8 bit palette / 24 bit color depth supported
	Bit 5	FALSE	8 bit palette / 16 bit color depth unsupported
		TRUE	8 bit palette / 16 bit color depth supported
	Bit 6	FALSE	4 bit 16 color unsupported
		TRUE	4 bit 16 color supported
	Bit 7	FALSE	1 bit monochrome unsupported
		TRUE	1 bit monochrome supported

Flags

Reserved for future extensions.

Basis datatype	Bit-Nr.	Code	Description
Boolean	Bit 0..7	–	reserved

2.8.3 ColorModes

Sets or retrieves the color mode the display works in. The mode parameter is related to the corresponding bits in the ColorModes member of the PhysicalMetrics property.

Function class: Enum

FBlock	FktID	OPType	Parameter
GraphicDisplay (0x60)	ColorModes (0x201)	GetInterface	
		Set	Mode
		SetGet	Mode
		Get	
		Status	Mode
		Interface	Flags, Class, Name, NMax, Class, OPTypes, Name, Units, DataType, Exponent, Min, Max, Step
		Error	ErrorCode, ErrorInfo

Parameter

Mode

Current color mode.

Basis datatype	Value	Description
Enum	0	TrueColor 32 bit
	1	TrueColor 24 bit
	2	HighColor 16 bit
	3	8 bit palette / 32 bit color depth
	4	8 bit palette / 24 bit color depth
	5	8 bit palette / 16 bit color depth
	6	4 bit 16 color
	7	1 bit monochrome

2.8.4 AvailableResources

Contains information about the total and available memory resources for graphical data. The numbers are given in bytes. As additional information, the memory sizes of the most important data structures are given, so the application can estimate the number of remaining graphical elements. Since some data structures may vary dynamically in size, the display's implementers should use reasonable median values in such cases.

Function class: Array of Record of { Number Number Number Number Number }

FBlock	FktID	OPType	Parameter
GraphicDisplay (0x60)	AvailableResources (0x202)	GetInterface	
		Get	Pos
		Status	Pos, Data
		Interface	Flags, Class, Name, NMax, Class, OPTypes, Name, Units, DataType, Exponent, Min, Max, Step
		Error	ErrorCode, ErrorInfo

Parameter

Pos

The parameter Pos={x,y} consists of Bytes and indicates the parameters to be retrieved and/or modified. Since the construct has only one dimension, the second byte remains unused.

Basis datatype	Exp	Range	Step	Unit
Unsigned Word	0	full range	1	none

Data

The contents of Data depends on the setting of the Pos parameter.

Basis datatype	Length	Description	
Stream	-	Pos	Data
		{x=0, y=0}	{ MemTotal, MemAvail, ContainerSize, ElementSize, VectorSize, Viewports, Palettes}

MemTotal

Total memory reserved for graphics data.

Basis datatype	Exp	Range	Step	Unit
Unsigned Long	0	full range	1	none

MemAvail

Memory currently available for graphics data.

Basis datatype	Exp	Range	Step	Unit
Unsigned Long	0	full range	1	none

ContainerSize

Number of bytes that is needed to store a container (without its elements).

Basis datatype	Exp	Range	Step	Unit
Unsigned Word	0	full range	1	none

ElementSize

Number of bytes that is needed to store an element (without its vectors).

Basis datatype	Exp	Range	Step	Unit
Unsigned Word	0	full range	1	none

VectorSize

Number of bytes that is needed to store a vector.

Basis datatype	Exp	Range	Step	Unit
Unsigned Word	0	full range	1	none

Viewports

Number of Viewports that can be created in addition to existing ones.

Basis datatype	Exp	Range	Step	Unit
Unsigned Byte	0	full range	1	none

Palettes

Number of palettes that can be created in addition to existing ones.

Basis datatype	Exp	Range	Step	Unit
Unsigned Byte	0	full range	1	none

2.8.5 CreateViewport

Creates a Viewport on the display. The socket data connection must be set up separately as the originator of the CreateViewport call may not be the FBlock to connect with.

Function class: Unclassified method

FBlock	FktID	OPType	Parameter
GraphicDisplay (0x60)	CreateViewport (0x203)	StartResult	Width, Height
		Result	Handle
		Processing	
		Error	ErrorCode, ErrorInfo

Parameter

Width

The width of the requested Viewport.

Basis datatype	Exp	Range	Step	Unit
Unsigned Word	0	full range	1	none

Height

The height of the requested Viewport.

Basis datatype	Exp	Range	Step	Unit
Unsigned Word	0	full range	1	none

Handle

Handle of the Viewport. By recommendation, this should be the connection handle of the socket data connection.

Basis datatype	Exp	Range	Step	Unit
Unsigned Long	0	full range	1	none

2.8.6 DestroyViewport

Destroys a Viewport on the display. The socket connection must be terminated separately.

Function class: Unclassified method

FBlock	FktID	OPType	Parameter
GraphicDisplay (0x60)	DestroyViewport (0x204)	StartResult	Handle
		Result	Handle
		Processing	
		Error	ErrorCode, ErrorInfo

Parameter

Handle

Handle of the Viewport. By recommendation, this should be the connection handle of the socket data connection.

Basis datatype	Exp	Range	Step	Unit
Unsigned Long	0	Full range	1	none

2.8.7 CreatePalette

Creates a color palette that is global for all Viewports.

Function class: Unclassified method

FBlock	FktID	OPType	Parameter
GraphicDisplay (0x60)	CreatePalette (0x205)	StartResult	Name
		Result	Handle
		Processing	
		Error	ErrorCode, ErrorInfo

Parameter

Name

The name of the palette. Must be different from those of already existing palettes.

Basis datatype	Max size
String	11

Handle

Handle of the palette. On failure, the display returns 0.

Basis datatype	Exp	Range	Step	Unit
Unsigned Byte	0	full range	1	none

2.8.8 DestroyPalette

Destroys a palette.

Function class: Unclassified method

FBlock	FktID	OPType	Parameter
GraphicDisplay (0x60)	DestroyPalette (0x206)	StartResult	Handle
		Result	Handle
		Processing	
		Error	ErrorCode, ErrorInfo

Parameter

Handle

Handle of the palette. On failure, the display returns 0.

Basis datatype	Exp	Range	Step	Unit
Unsigned Byte	0	full range	1	none

2.8.9 PaletteEntries

Retrieves or modifies palette entries.

Function class: Array of Array of Number

FBlock	FktID	OPType	Parameter
GraphicDisplay (0x60)	PaletteEntries (0x207)	GetInterface	
		Set	Pos, Data
		Get	Pos
		SetGet	Pos, Data
		Status	Pos, Data
		Interface	Flags, Class, Name, NMax, Class, OPTypes, Name, Units, DataType, Exponent, Min, Max, Step
		Error	ErrorCode, ErrorInfo

Parameter

Pos

The parameter Pos={x,y} consists of two Bytes and indicates the parameter to be retrieved and/or modified. The first byte denotes the palette handle, the second the entry index

Basis datatype	Exp	Range	Step	Unit
Unsigned Word	0	full range	1	none

Data

The contents of Data depends on the setting of the Pos parameter.

Basis datatype	Length	Description	
Stream	-	Pos	Data
		{palette, index}	Color[palette][index]

Color

Color entry of the form Red, Green, Blue, Alpha (MSB to LSB).

Basis datatype	Exp	Range	Step	Unit
Unsigned Long	0	full range	1	none

2.8.10 PaletteCount

Returns the number of existing palettes.

Function class: Number

FBlock	FktID	OPType	Parameter
GraphicDisplay (0x60)	PaletteCount (0x208)	GetInterface	
		Get	
		Status	Count
		Interface	Flags, Class, Name, NMax, Class, OPTypes, Name, Units, DataType, Exponent, Min, Max, Step
		Error	ErrorCode, ErrorInfo

Parameter

Count

Number of existing palettes.

Basis datatype	Exp	Range	Step	Unit
Unsigned Long	0	full range	1	none

2.8.11 PaletteInformation

Function class: Array of Record of { Number Number String }

FBlock	FktID	OPType	Parameter
GraphicDisplay (0x60)	PaletteInformation (0x209)	GetInterface	
		Get	Pos
		Status	Pos, Data
		Interface	Flags, Class, Name, NMax, Class, OPTypes, Name, Units, DataType, Exponent, Min, Max, Step
		Error	ErrorCode, ErrorInfo

Parameter

Pos

The parameter Pos={x,y} consists of two Bytes and indicates the parameters to be retrieved and/or modified.

Basis datatype	Exp	Range	Step	Unit
Unsigned Word	0	full range	1	none

Data

The contents of Data depends on the setting of the Pos parameter.

Basis datatype	Length	Description	
Stream	-	Pos	Data
		{x=n, y=0}	{ Handle[n], Used[n], Name[n] }

Handle

Handle of the palette.

Basis datatype	Exp	Range	Step	Unit
Unsigned Byte	0	1..255	1	none

Used

1. Number of used colors in the palette. 2. First unused index.

Basis datatype	Exp	Range	Step	Unit
Unsigned Byte	0	full range	1	none

Name

The name of the palette.

Basis datatype	Max size
String	11

2.8.12 GraphicData

This function is used to transfer graphic data to (Set OPType) and from (Status OPType) the device via MOST High Protocol. All messages correspond to a certain Viewport that is identified through the Handle parameter. Status messages are sent only to the DeviceID created the respective Viewport through the CreateViewport method. The notification mechanism is not used for this property.

Function class: Array of Record of { Number Number String }

FBlock	FktID	OPType	Parameter
GraphicDisplay (0x60)	GraphicData (0x20A)	GetInterface	
		Get	Pos
		Status	Pos, Data
		Interface	Flags, Class, Name, NMax, Class, OPTypes, Name, Units, DataType, Exponent, Min, Max, Step
		Error	ErrorCode, ErrorInfo

Parameter

Data

Graphic data like specified in section 2.2: Protocol Structures

Basis datatype	Length	Description
Stream	-	data

3 FunctionBlock Definition

3.1 1 GraphicDisplay (FBlockID=0x60)

3.1.1 FktIDs (0x000)

With the property FktIDs the functions of a function block may be inquired.

3.1.1.1 Format of Function

Function classes: Unclassified Property

FBlock	Function	OPType	Parameter
GraphicDisplay (0x60)	FktIDs (0x000)	Get	
		Status	BitField
		Error	ErrorCode, ErrorInfo

3.1.1.2 Parameter

BitField

RLE-coded bitfield of available functions Remark: FktIDs are 12 Bit encoded !

Basis datatype	Length	Description
Stream		FktID1, FktID2, ...

3.1.2 Notification (0x001)

This property administrates the Notification Matrix of a function block.

3.1.2.1 Format of Function

Function classes: Unclassified Property

FBlock	Function	OPType	Parameter
GraphicDisplay (0x60)	Notification (0x001)	Set	Control, DeviceID, FktIDList
		Get	FktID
		Status	FktID, DeviceIDList
		Error	ErrorCode, ErrorInfo

3.1.2.2 Parameter

Control

The parameter Control determines, where the entry has to be done, or the deletion respectively. SetAll = Entry of DeviceID in all properties that support Notification SetFunction = Entry of DeviceID for the specified functions in the Notification-Matrix ClearAll = Deletion of DeviceID at all functions of the Notification-Matrix ClearFunction = Deletion of DeviceID for the specified functions in the Notification-Matrix

Basis datatype	Range of values	Code	Description
Enum	0x00..0x03	0x00	SetAll
		0x01	SetFunction
		0x02	ClearAll
		0x03	ClearFunction

DeviceID

Rx/TxLog of a device or group address

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

FktID

Function

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

DeviceIDList

List of Devices

Basis datatype	Length	Description
Stream		DeviceID {, DeviceID}

FktIDList

List of functions with a maximum of 4.

Basis datatype	Length	Description
Stream	8	FktID {, FktID}

3.1.3 NotificationCheck (0x002)

Under certain system conditions it can be helpfull if a device can check whether its entries are still existent in the notification matrix or not. In case of error, a device is able to renew its entries.

3.1.3.1 Format of Function

Function classes: Unclassified Property

FBlock	Function	OPType	Parameter
GraphicDisplay (0x60)	NotificationCheck (0x002)	Get	DeviceID
		Status	DeviceID, FktIDList
		Error	ErrorCode, ErrorInfo

3.1.3.2 Parameter

FktIDList

List of functions.

Basis datatype	Length	Description
Stream	-	FktID {, FktID}

FktID

Function

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

DeviceID

Rx/TxLog of a device or groupaddress

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

3.1.4 PhysicalMetrics (0x200)

PhysicalMetrics contains a description of the display’s hardware properties like resolution, size and color depth.

3.1.4.1 Format of Function

Function classes: Record of { Number Number Number Number BitField BitField }

FBlock	Function	OPType	Parameter
GraphicDisplay (0x60)	PhysicalMetrics (0x200)	Get	Pos
		Status	Pos, Data
		Error	ErrorCode, ErrorInfo

3.1.4.2 Parameter

Data

The content of Data depends on parameter Pos={x,y}.

Basis datatype	Length	Description	
Stream	-	Pos	Data
		{ x=0, y=0 }	{ XRes, YRes, XSize, YSize, ColorModes, Flags}
		{ x=1, y=0 }	XRes
		{ x=2, y=0 }	YRes
		{ x=3, y=0 }	XSize
		{ x=4, y=0 }	YSize
		{ x=5, y=0 }	ColorModes
		{ x=6, y=0 }	Flags

XRes

Horizontal display resolution in pixels.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	0	full range	1	none

YRes

Vertical display resolution in pixels.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	0	full range	1	none

XSize

Horizontal display size in millimeters.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	-3	full range	1	m

YSize

Vertical display size in millimeters.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	-3	full range	1	m

ColorMode

Available color modes.

Basis datatype	Bit #	Code	Description
Boolean	Bit 0	False	TrueColor 32 bit unsupported
		True	TrueColor 32 bit supported
	Bit 1	False	TrueColor 24 bit unsupported
		True	TrueColor 24 bit supported
	Bit 2	False	HighColor 16 bit unsupported
		True	HighColor 16 bit supported
	Bit 3	False	8 bit palette / 32 bit color depth unsupported
		True	8 bit palette / 32 bit color depth supported
	Bit 4	False	8 bit palette / 24 bit color depth unsupported
		True	8 bit palette / 24 bit color depth supported
	Bit 5	False	8 bit palette / 16 bit color depth unsupported
		True	8 bit palette / 16 bit color depth supported
	Bit 6	False	4 bit 16 color unsupported
		True	4 bit 16 color supported
	Bit 7	False	1 bit monochrome unsupported
		True	1 bit monochrome supported

Flags

Reserved for future use.

Basis datatype	Bit #	Code	Description
Boolean	Bit 0 ... 7	-	reserved

Pos

The parameter Pos={x,y} consists of Bytes and indicates the parameters to be retrieved and/or modified. Since the construct has only one dimension, the second byte remains unused.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	0	full range	1	none

3.1.5 ColorMode (0x201)

Sets or retreives the color mode the display works in. The mode parameter is related to the corresponding bits in the ColorModes member of the PhysicalMetrics property.

3.1.5.1 Format of Function

Function classes: Enumeration

FBlock	Function	OPType	Parameter
GraphicDisplay (0x60)	ColorMode (0x201)	Set	Mode
		Get	
		SetGet	Mode
		Status	Mode
		Error	ErrorCode, ErrorInfo

3.1.5.2 Parameter

Mode

Current color mode.

Basis datatype	Range of values	Code	Description
Enum	0x00..0x07	0x00	TrueColor 32 bit
		0x01	TrueColor 24 bit
		0x02	HighColor 16 bit
		0x03	8 bit palette / 32 bit color depth
		0x04	8 bit palette / 24 bit color depth
		0x05	8 bit palette / 16 bit color depth
		0x06	4 bit 16 color
		0x07	1 bit monochrome

3.1.6 AvailableResources (0x202)

Contains information about the total and available memory resources for graphical data. The numbers are given in bytes. As additional information, the memory sizes of the most important data structures are given, so the application can estimate the number of remaining graphical elements. Since some data structures may vary dynamically in size, the display's implementers should use reasonable median values in such cases.

3.1.6.1 Format of Function

Function classes: Record of { Number Number Number Number Number Number Number }

FBlock	Function	OPType	Parameter
GraphicDisplay (0x60)	AvailableResources (0x202)	Get	
		Status	Pos, Data
		Error	ErrorCode, ErrorInfo

3.1.6.2 Parameter

Data

The contents of Data depends on the setting of the Pos parameter.

Basis datatype	Length	Description	
Stream	-	Pos	Data
		{ x=0, y=0 }	{ MemTotal, MemAvail, ContainerSize, ElementSize, VectorSize, Viewports, Palettes}
		{ x=1, y=0 }	MemTotal
		{ x=2, y=0 }	MemAvail
		{ x=3, y=0 }	ContainerSize
		{ x=4, y=0 }	ElementSize
		{ x=5, y=0 }	VectorSize
		{ x=6, y=0 }	Viewports
		{ x=7, y=0 }	Palettes

MemTotal

Total memory reserved for graphics data.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Long	0	full range	1	none

MemAvail

Memory currently available for graphics data

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Long	0	full range	1	none

ContainerSize

Number of bytes that is needed to store a container (without its elements).

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	0	full range	1	none

ElementSize

Number of bytes that is needed to store an element (without its vectors).

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	0	full range	1	none

VectorSize

Number of bytes that is needed to store a vector.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	0	full range	1	none

Viewports

Number of viewports that can be created in addition to existing ones.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Byte	0	full range	1	none

Palettes

Number of palettes that can be created in addition to existing ones.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Byte	0	full range	1	none

Pos

The parameter Pos={x,y} consists of Bytes and indicates the parameters to be retrieved and/or modified. Since the construct has only one dimension, the second byte remains unused.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	0	full range	1	none

3.1.7 CreateViewport (0x203)

Creates a viewport on the display. The socket data connection must be set up separately by opening a MOSThigh connection to the GraphicData property.

3.1.7.1 Format of Function

Function classes: Unclassified Method

FBlock	Function	OPType	Parameter
GraphicDisplay (0x60)	CreateViewport (0x203)	Processing	
		Result	Handle
		StartResult	Width, Height
		Error	ErrorCode, ErrorInfo

3.1.7.2 Parameter

Handle

Handle of the viewport.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Long	0	full range	1	none

Height

The height of the requested viewport.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	0	full range	1	none

Width

The width of the requested viewport.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	0	full range	1	none

3.1.8 DestroyViewport (0x204)

Destroys a viewport on the display. The socket connection must be terminated separately.

3.1.8.1 Format of Function

Function classes: Unclassified Method

FBlock	Function	OPType	Parameter
GraphicDisplay (0x60)	DestroyViewport (0x204)	Result	
		StartResult	Handle
		Error	ErrorCode, ErrorInfo

3.1.8.2 Parameter

Handle

Handle of the viewport.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Long	0	full range	1	none

3.1.9 CreatePalette (0x205)

Creates a color palette that is global for all viewports.

3.1.9.1 Format of Function

Function classes: Unclassified Method

FBlock	Function	OPType	Parameter
GraphicDisplay (0x60)	CreatePalette (0x205)	Result	Handle
		StartResult	Name
		Error	ErrorCode, ErrorInfo

3.1.9.2 Parameter

Handle

Handle of the palette. On failure, the display returns 0.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Byte	0	full range	1	none

Name

The name of the palette. Must be different from those of already existing palettes.

Basis datatype	MaxSize
String	11

3.1.10 DestroyPalette (0x206)

Destroys a palette.

3.1.10.1 Format of Function

Function classes: Unclassified Method

FBlock	Function	OPType	Parameter
GraphicDisplay (0x60)	DestroyPalette (0x206)	Result	Handle
		StartResult	Handle
		Error	ErrorCode, ErrorInfo

3.1.10.2 Parameter

Handle

Handle of the palette. On failure, the display returns 0.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Byte	0	full range	1	none

3.1.11 PaletteEntries (0x207)

Retrieves or modifies palette entries.

3.1.11.1 Format of Function

Function classes: Array of { Number }

FBlock	Function	OPType	Parameter
GraphicDisplay (0x60)	PaletteEntries (0x207)	Set	Pos, Data
		Get	Pos
		SetGet	Pos, Data
		Status	Pos, Data
		Error	ErrorCode, ErrorInfo

3.1.11.2 Parameter

Data

The contents of Data depends on the setting of the Pos parameter.

Basis datatype	Length	Description	
Stream	-	Pos	Data
		{ x=0, y=0 }	{Color[1][1], Color[1][2], ... Color[NmaxX][NmaxY]}
		{ x>0, y=0 }	{Color[x][1], Color[x][2], ... Color[x][NmaxY]}
		{ x>0, y>0 }	Color[x,y]

Color

No

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Long	0	full range	1	none

Pos

The parameter Pos={x,y} consists of two Bytes and indicates the parameter to be retrieved and/or modified. The first byte denotes the palette handle, the second the entry index

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	0	full range	1	none

3.1.12 PaletteCount (0x208)

Returns the number of existing palettes.

3.1.12.1 Format of Function

Function classes: Number

FBlock	Function	OPType	Parameter
GraphicDisplay (0x60)	PaletteCount (0x208)	Get	
		Status	Count
		Error	ErrorCode, ErrorInfo

3.1.12.2 Parameter

Count

Number of existing palettes.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Byte	0	full range	1	none

3.1.13 PaletteInformation (0x209)

NoDescription

3.1.13.1 Format of Function

Function classes: Array of { Record of { Number Number String } }

FBlock	Function	OPType	Parameter
GraphicDisplay (0x60)	PaletteInformation (0x209)	Get	Pos
		Status	Pos, Data
		Error	ErrorCode, ErrorInfo

3.1.13.2 Parameter

Data

Basis datatype	Length	Description	
Stream	-	Pos	Data
		{ x=0, y=0 }	{Handle[1], Used[1], Name[1], ... Handle[Nmax], Used[Nmax], Name[Nmax]}

		{ x>0, y=0 }	{Handle[x], Used[x], Name[x]}
		{ x>0, y=1 }	Handle[x]
		{ x>0, y=2 }	Used[x]
		{ x>0, y=3 }	Name[x]

Handle

Handle of the palette.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Byte	0	full range	1	none

Used

---

1. Number of used colors in the palette. All indexes from 1 up to Used are considered as "used". Hence, Used+1 represents the first unused index.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Byte	0	full range	1	none

Name

The name of the palette.

Basis datatype	MaxSize
String	11

Pos

The parameter Pos={x,y} consists of two Bytes and indicates the parameters to be retrieved and/or modified.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Word	0	full range	1	none

3.1.14 GraphicData (0x20A)

This function is used to transfer graphic data to (Set OpType) and from (Status OpType) the device via MOSThigh. All messages correspond to a certain viewport that is identified through the Handle parameter. Status messages are sent only to the DeviceId created the respective viewport through the CreateViewport method. The notificaiton mechanism is not used for this property.

3.1.14.1 Format of Function

Function classes: Unclassified Property

FBlock	Function	OPType	Parameter
GraphicDisplay (0x60)	GraphicData (0x20A)	Set	Handle, Data
		Status	Handle, Data
		Error	ErrorCode, ErrorInfo

3.1.14.2 Parameter

Data

Graphic Data like specified in the MOST Function Block GraphicDisplay Specification.

Basis datatype	Length	Description
Stream	-	No

Handle

The viewport handle.

Basis datatype	Exp.	Range of values	Step	Unit
Unsigned Long	0	full range	1	none

4 FunctionBlock Dynamic Specification