Proteus-V SCS
Software Communication Specification

Version V3.3
April 13, 2023

Table of Contents

1. Communication

1.1 Command Format

All commands start with $VL followed by two-character command ID, a comma, payload (command specific parameters), an asterisk, two-character checksum and new line character.

Start	Command ID	,	Payload	*	Checksum	End
$VL	**ii**	**,**	**Command specific parameters**	*****	**CS**	**LF CR**
$VL	07	,	400,200,300,50,0,33	*	1D	0A 0D
$VL	07	,	400,200,300,50,0,33	*	**XX**	0A 0D

An example is shown below:

```
“$VL07,400,200,300,50,0,33*1D\r\n”       // Command ID=07,    checksum included (1D)
“$VL07,400,200,300,50,0,33*XX\r\n”       // Command ID=07, No checksum included (XX)
```

1.2 Checksum Computation

The checksum field is the last field in a sentence and follows the checksum delimiter character “*”. The checksum is the 8-bit exclusive OR of all characters in the sentence, including “,” delimiters, between but not including the “$” and the “*” delimiters. The hexadecimal values of the most significant and least significant 4 bits of the result is converted to two ASCII characters (0-9, A-F (upper case)) for transmission. The most significant character is transmitted first. Example: “$GPGLL,5057.970,N,00146.110,E,142451,A*27\r\n”

In C checksum computation would be written as:

```
char sentence [] = “GPGLL,5057.970,N,00146.110,E,142451,A”;
int i;
char checksum = 0;

for ( i = 0; i < strlen(sentence); i++)
       checksum ^= sentence[i];
```

Although not recommended, checksum can be excluded by replacing it with **XX.**

1.3 Reply Format

The only commands that require a formatted reply are `$VL24, $VL30, $VL35, $VL42`.

As an example, reply to the command `$VL24` is shown below:

Start	Command ID	,	Reply	,	Checksum	End
`<STX>`	`ii`	`,`	`Reply`	`,`	`cc`	`<ETX>`
`02`	`24`	`,`	`01234567`	`,`	`89`	`03`

Reply to command `$VL24` in HEX	→	02	32	34	2C	30	31	32	33	34	35	36	37	2C	37	38	03
Reply to command `$VL24` in ASCII	→	STX	2	4	,	0	1	2	3	4	5	6	7	,	8	9	ETX

For the remaining commands, Proteus returns a single byte (i.e., ACK or NAK) to indicate whether command was accepted or rejected.

Reply character	Description
6 (ACK)	Message was accepted
0 (NAK)	Message was rejected

1.4 CSV FORMATS

A CSV is an ASCII sentence composed of a unique header, followed by up to 12 comma separated values and a checksum.

```
$Header,VAL1,VAL2,VAL3,VAL4,VAL5,VAL6,VAL7,VAL8,VAL9,VAL10,VAL11,VAL12*CS
```

$	Signifies start of the sentence.
Header	Sentence header. See command $VL29 to set unique sentence header.
VALn	Each sentence contains multiple values (VALn) delimited by commas.
*****	The asterisk serves as checksum delimiter.
CS	The checksum field contains two ASCII characters which indicate the hexadecimal value of the checksum.

PROTEUS supports 4 different CSV (Comma Separate Values) and 2 different SSV (Space Separate Values) sentences:

Type	Sentence includes	Sentence Structure	Example	Location of parsed VALn
CSV1	$Header, Values…, Checksum	`$HEADER,VAL1,VAL2,VAL3,...VALn*CS`	`$STEVE,45,315,200,100*XX`	`In sentence A,B,C,D`
CSV2	$Header, Values…	`$HEADER,VAL2,VAL3,...`	`$BRIAN,45,315,200,100`	`In sentence A,B,C,D`

CSV3
SSV3
	$Values,..
	$Values ...
		`$VAL1,VAL2,VAL3,...`
		`$VAL1 VAL2 VAL3 ...`
			`$45,315,200,100`
			`$45 315 200 100`
				`In sentence A`
				`In sentence A`

CSV4
SSV4
	Values,...
	Values ...
		`VAL1,VAL2,VAL3,..`
		`VAL1 VAL2 VAL3 ..`
			`45,315,200,100`
			`45 315 200 100`
				`In sentence A`
				`In sentence A`

Upon reception of a **CSV** sentence and confirmation of the sentence header (only CSV1), PROTEUS parses the sentence. Parsed values (VAL*1 ... VAL12*) are sequentially stored in Registers # 40 through 87. Any widgets linked to these registers will automatically get updated. CSV sentences vary in length, but each VALn is limited to 40 characters or less. For more detail see "Display Text Via RS232/Ethernet/USB" in the User Manual.

1.5 RS232 PORTS

1.6 COM1

COM1 (DB9) is configured as DTE (PC) i.e., RX=Pin2, TX=Pin3. Thus, sensors such as GPS can be directly connected to the DB9 without the need for NULL modem cable. However, when using COM1, a NULL modem cable is required to interface PROTEUS to a PC.

This port can be used to receive commands defined in this document or connect any RS232 sensor/device that transmits *CSV1, CSV2, CSV3, CSV4, SSV3, SSV4* type data formats. For more detail see "Display Text Via RS232/Ethernet/USB" in the User Manual.

1.7 COM2

COM2 is located internal. Signals TX & RX are provided at J16 connector (Compatible with Garmin GPS 18x LVC) as well as Terminal Block J54.

This port can also be used to receive commands defined in this document or connect any RS232 sensor/device that transmits *CSV1, CSV2, CSV3, CSV4, SSV3, SSV4* type data formats. For more detail see "Display Text Via RS232/Ethernet/USB" in the User Manual.

1.8 MINI USB PORT

This port is USB Device CDC-ACM class. It allows a USB host (PC) to communicate with the device (Proteus) as a serial device.
There is no baud rate associated with this interface and transfer speed of 1.7 Mbit/s to 4.1Mbit/s can be achieved.

This port can be used to send commands defined in this document or any *CSV1 type data formats*. Parsed values from CSV1 sentences are stored in "CSV Sentence-A" parameters. For more detail see "Display Text Via RS232/Ethernet/USB" in the User Manual.

This port is also used for loading the firmware into FLASH.

1.9 ETHERNET PORT

This port can be used to send commands defined in this document or any *CSV1 type data formats*. Parsed values from CSV1 sentences are stored in “CSV Sentence-A” parameters. For more detail see “Display Text Via RS232/Ethernet/USB” in the User Manual.

- 10M/100M auto sensing network interface.
- Networking: Static or DHCP IPv4 addressing
- Subnet Mask: Configurable. Default 255.255.255.0,
- Default Gateway: 0.0.0.0
- UDP protocol. Port 9999
- Users should ignore ON/OFF state of the LED in the Ethernet Connector as they are not wired in the conventional manner.

Assume the following ten commands must be transmitted to Proteus every 10msec:

```
cmd1  = “$VL13,600,600,400,300,7,1,3,16,32,1,0,Lower Left*XX\n”
cmd2  = “$VL13,600,600,400,300,9,1,3,16,33,0,0,Lower Right*XX\n”
cmd3  = “$VL13,600,600,400,300,3,1,3,16,33,0,0,Upper Right*XX\n”
cmd4  = “$VL13,600,600,  0,  0,0,1,3,16,32,0,0,Upper Left*XX\n”
cmd5  = “$VL13,600,600,400,300,5,3,3,16,32,0,0,Center*XX\n”
cmd6  = “$VL52,100,100,300,50,300,300,5,1,1,5*XX\n”
cmd7  = “$VL55,400,100,1000,300,28,1,1,4*XX\n”
cmd8  = “$VL54,300,300,200,-45,15,29,1*XX\n”
cmd9  = “$VL53,400,475,100,-180,20,20,30,1,1,0*XX\n”
cmd10 = ”$VL25,10,16,16*XX\n”
```

Option 1
Send each command separately i.e., use ten UDP packets: UDP (cmd1) + UDP (cmd2) + UDP (cmd3) + UDP (cmd4) + UDP (cmd5) + UDP (cmd6) + UDP (cmd7) + UDP (cmd8) + UDP (cmd9) + UDP (cmd10) Proteus will respond with 10 ACK

Option 2
Concatenate ten commands into one string and send via one UDP packet: UDP (cmd1+cmd2+cmd3+cmd4+cmd5+cmd6+cmd7+cmd8+cmd9+cmd10) Proteus will respond with 10 ACK. *Note, keep number commands to 200 or less and the length of the concatenated string to 1536 bytes or less.*

There is no benefit in updating parameters any faster than the video frame rate i.e., screen refresh rate. For example, when using 1080p @30Hz which is the fastest frame rate Proteus can handle, keep parameter update rate to 30Hz or less. The typical parameter refresh rate is 10Hz.

2. Video processing

In order to preserve the original video quality, Proteus does not *capture* or *scale* video. The Video input HD or SD maintains its *original resolution and quality*. All OSD functions are superimposed into the video "on-the-fly". Therefore, the delay from the video input to the video output is 18 cycles (242 nsec) of 74.25MHz.

3. Video Frame Buffer and Layers

Proteus has 256M of Frame Buffer which is partitioned into four overlapping OSD layers. OSD layer '2' is 6 times larger than a single HD video frame (shown as penguin) and OSD layer '3' is 2 times larger. By default, page 0 of each layer is made visible. The additional pages (in layer 2 and 3) can be made visible by using command VL03.

Immediately after power-up, all PNG images are automatically copied from FLASH into Page 1 of Layer 3. All fonts are also copied from FLASH into Page 1 of Layer 2. Proteus uses BitBlt to copy these objects (PNG & Fonts) from Pages 1 into the visible screens.

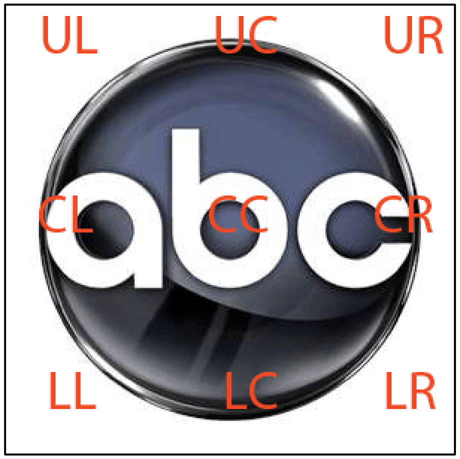
		Upper Left Corner ↓	Layer 0	Layer 1	Layer 2	Layer 3
			(1920 x 1024)	(1920 x 1024)	(1920 x 1080)	(1920 x 1080)
			Index	Index	Index	ARGB8888
			Use for: Text + Drawing	Use for: Text + Drawing	Use for: Text + Drawing	Use for: Text + Drawing + **PNG**
Page	0	*0,0*	Page0	Page0	Page0 visible by default	Page0 visible by default
	1	*0,1080*	N/A	N/A	Page1. Used internally	Page1. Used internally
	2	*0,2160*	N/A	N/A	Page2 Available	N/A
	3	*0,3240*	N/A	N/A	Page3 Available	N/A
	4	*0,4320*	N/A	N/A	Page4 Available	N/A
	5	*0,5400*	N/A	N/A	Page5 Available	N/A

When drawing objects on pages 2..5, make sure x,y coordinates are within the page boundary. Failure may result in unexpected artifact on the screen. On Layer 2, to display Page 3, press Alt+Ctrl+Shift+2 three times. On Layer 3, to display page 1 press Alt+Ctrl+Shift+3 once. Pressing ESC key will display to Page 0.

4. Text Justification

Some commands provide the capability to justify texts within a rectangular area or graphic object. This is done by defining the "Justify" parameter in the command:

Justify	Description	
'1'	*Upper Left.*	*UL*
'2'	*Upper Center.*	*UC*
'3'	*Upper Right.*	*UR*
'4'	*Center Left.*	*CL*
'5'	*Center Center*	*CC*
'6'	*Center Right.*	*CR*
'7'	*Lower Left.*	*LL*
'8'	*Lower Center.*	*LC*
'9'	*Lower Right.*	*LR*



5. Layer

Commands such as *Image:Display*, *String:Draw*, *Rect:Copy* operate on a specific layer. This is done by defining the "Dst" or "Src" parameter in each command:

Dst or Src	Description
'0'	Operate on Layer 0
'1'	Operate on Layer 1
'2'	Operate on Layer 2
'3'	Operate on Layer 3

6. Commands

6.1 Layer

6.1.1 Clear Layer

This command erases the entire content of specified layer.

Cmnd ID	Payload
`01`	`Layer`

Parm	Description	Range
`Layer`	0 = Layer 0 1 = Layer 1 2 = Layer 2 3 = Layer 3 4 = All layers	Layers
Example		
`$VL01,0*XX`	*// Clear layer 0*	
`$VL01,4*XX`	*// Clear all layers*	

6.1.2 Set Layer Priority

This command set the layer priority.

Cmnd ID	Payload	
`02`	`Layer`	`Priority`

Parm	Description	Range
`Layer`	0..3 = Layer 0..3 4 = Background	*0..4*
`Priority`	0 = Priority 5 (Lowest priority) 1 = Priority 4 2 = Priority 3 3 = Priority 2 4 = Priority 1 (Highest priority)	*0..4*
Example		
`$VL02,2,4*XX`	*// Set Layer 2 to highest priority*	

6.1.3 Select Page

Layer 2 supports 6 pages and layer 3 supports 2 pages. On each layer, only one page can be displayed at a time. This command provides a mean to make alternative pages visible.

Cmnd ID	Payload	
03	Layer	Page

Parm	Description	Range
Layer	Select page 2 or 3	*2..3*
Page	Make this page visible	*0..5*
Example		
$VL03,2,1*XX	// Make Page 1 of Layer 2 visible	
$VL03,2,5*XX	// Make Page 5 of Layer 2 visible	
$VL03,3,1*XX	// Make Page 1 of Layer 3 visible	

6.1.4 Set Layer Alpha Blend

This command set the alpha blending mode and value of specified layer.

Cmnd ID	Payload		
05	Layer	Alpha mode	Alpha Value MX[5:0]

Parm	Description	Range
Layer	0..3 = Layer 0..3 4 = Background	*0..4*
Alpha mode	0 = OFF 1 = Use Alpha Value of Pixel Data AMX[5:0] 2 = Use Alpha value of Layer MX[5:0] 3 = Use MX[5:0] * AMX[5:0]	*0..3*
Alpha value	Layer Alpha value	*0..63*
Example		
$VL05,1, 1, 63*XX		

Alpha Value	Pixel Alpha Rendering	
0	100% Graphics	0 % Video
31	50% Graphics	50 % Video
63	0% Graphics	100 % Video

6.2 Set Color Index

Assign a color to a specified index within color palette.

Cmnd ID	Payload	
04	Index	Color

Parm	Description	Range
Index	Color Index	0..255
Color	aaRRGGBB where aa is alpha blend. Only most significant 6-bits of ‘aa’ are used. Least significant 2 bits are ignored i.e. 0x00..0xFC	aaRRGGBB
Example		

`$VL04,16,FC00FF00*XX          // Set palette index 35 to GREEN. Set alpha-blend to FC (opaque)`
`$VL04,75,7C0000FF*XX          // Set palette index 75 to BLUE.  Set alpha-blend to mid-transparency 7C`		

Color indexes 0, 16..47 & 70..79 are used by Proteus and best not to alter them. The remaining color indexes can be modified. Set the color index to your desired color prior to using it in subsequent commands.

Table below shows default color assigned to index 0..15 that can be used with any command.

Index	Color		
0		Transparent	0x00000000
1	■	Black	0xFC010101
2		White	0xFCFFFFFF
3	■	Red	0xFCFF0000
4	■	Green	0xFC00FF00
5	■	Blue	0xFC0000FF
6	■	Yellow	0xFCFFFF00
7	■	Cyan	0xFC00FFFF
8	■	Magenta	0xFCFF00FF
9	■	Silver	0xFCC0C0C0
10	■	Gray	0xFC808080
11	■	Maroon	0xFC800000
12	■	Olive	0xFC808000
13	■	Green	0xFC008000
14	■	Purple	0xFC800080
15	■	Teal	0xFC008080

6.3 FONT COLOR

Proteus provides 8 different fonts stored in FLASH memory. At power up, all 8 fonts are also copied into VRAM. Fonts loaded in FLASH are assigned ID 0..7 and same fonts loaded in VRAM are assigned ID 8..15. In another word, FONT ID1 is same as FONT ID9.

The difference between the two group is the drawing speed and how colors are applied to the font. When using say font ID1, each text field (using font ID1) can have any unique color. However, when using font ID9, all texts fields (using font ID9) have the same color. The default colors for font ID 8..15 are set via color index 56..63 and can be changed via command `$VL04` if needed.

Fon ID 8..15 provide the fastest drawing speed (using BitBlt) and are preferred.

	Fonts in FLASH			Fonts in VRAM	
Font File	ID	Can be drawn with Color-Index		ID	Can be drawn with Color-Index
Fonts\Verdana12.fnt	0	1..255		8	56
Fonts\Verdana16.fnt	1	1..255		9	57
Fonts\Verdana18.fnt	2	1..255		10	58
Fonts\Verdana22.fnt	3	1..255		11	59
Fonts\Verdana28.fnt	4	1..255		12	60
Fonts\Verdana34.fnt	5	1..255		13	61
Fonts\Verdana39.fnt	6	1..255		14	62
Fonts\Verdana44.fnt	7	1..255		15	63

To change font ID9 color to white, issue the following command: `$VL04,57,FCFFFFFF*XX`

To change font ID9 color to red, issue the following command: `$VL04,57,FCFF0000*XX`

6.4 Rectangle area

6.4.1 Copy area

Copy a rectangular area from a source location (Sx, Sy) to a destination location (Dx, Dy).

Cmnd ID	Payload							
06	Sx	Sy	W	H	Dx	Dy	Dst Layer	Src Layer

Parm	Description	Range
Sx	Source location X	*0..1920*
Sy	Source location Y	*0..1080*
W	Width of Rectangle	*0..1920*
H	Height of Rectangle	*0..1080*
Dx	Destination location X	*0..1920*
Dy	Destination location Y	*0..1080*
Dst Layer	destination layer	Layers
Src Layer	Source layer	Layers
Example		
`$VL06,100,100,200,200,600,300,0,1*XX        // Copy Rect area (100,100,200,200) to (600,300) L1→L0`		

6.4.2 Erase area

6.4.3 Paint area

Paint or erase a rectangular area.

Cmnd ID	Payload					
07	Dx	Dy	W	H	Dst Layer	Color

Payloads	Description	Range
Dx	Destination location X	*0..1920*
Dy	Destination location Y	*0..1080*
W	Width of Rectangle	*0..1920*
H	Height of Rectangle	*0..1080*
Dst Layer	Destination Layer	Layers
Color	Fill color index. If index = 0, rectangle is erased	0..255
Example		

`$VL07,100,100,200,200,1,0*XX             // Erase area x=100,y=100,w=200.h=200 on layer 1`
`$VL07,100,100,200,200,0,30*XX            // Paint area x=100,y=100,w=200.h=200 on layer 0, using GREEN`		

6.5 Text

6.5.1 Store Text

Store frequently used texts in FLASH. Various commands use pre-stored texts by simply referencing a string ID.

Cmnd ID	Payload	
10	String ID	String

Payloads	Description	Range
String ID	*Assigned String ID*	*0..95*
String	*The space allocated to each string is 128 bytes. If string length exceeds 128 bytes, excess characters will be stored in ID+1 location.*	
Example		
$VL10,25,VideoLogix*XX	// Store 'VideoLogix' in FLASH and assign ID#25	

6.5.2 Delete Text

Delete one or more strings from FLASH.

Cmnd ID	Payload	
11	IDm	IDn

Payload	Description	Range
IDm	ID of the first string to delete	*0..95*
IDn	ID of the last string to delete	*0..95*
Example		
$VL11,25,25*XX $VL11,25,29*XX	// Delete string# 25 // Delete string# 25,26,27,28,29	

6.5.3 Get text width & height

This command computes the width & height of a string using a font.

Cmnd ID	Payload	
30	Font	String

Parm	Description	Range
font	Font ID	*0..15*
String	String provided in order to determine its width & height using font ID	*ASCII*
Command		
`$VL30,3,What is width & height of this text using font ID3*XX`		
Reply:		
`<STX>30,392,29,XX<ETX>            // Width = 392, Height = 29`		

6.5.4 Draw Text Direct

Use this command to draw text on the screen directly. Text can be justified within a rectangle area on layer 0..3. Rectangle area can be filled with an alpha-blended color (Bcolor).

Cmnd ID	Payload												
13	X	Y	W	H	Justify	Layer	Font	Bcolor	Fcolor	CLR	Text ID	Text (optional)	

13	X	Y	W	H	1..9 (9)	1	2	16	32	1	0	Hello	LR justify text in rect area X,Y,W,H (Figure 1)
13	X	Y	W	H	1..9 (5)	1	2	16	32	1	0	Hello	Center text (Figure 2)
13	X	Y	0	0	1 or 5 (1)	1	2	16	32	1	0	Proteus	Print text starting @ X,Y (Figure 3)
13	X	Y	0	0	1 or 5 (5)	1	2	16	32	1	0	Proteus	Print text starting @ X,Y (Figure 4)

Payload Parm	Description	Range
Area's X, Y, W, H	A rectangle area in which to justify the text. Rectangle area can be filled with color index Bcolor. If (W, H) = (0,0), there is no rectangle and text is drawn as shown in figure 3 or 4.	*0..1920* *0..1080* *0..1920* *0..1080*
Justify	Text justification works best when area's W & H is greater than text's W & H. *(if required, use command $VL30 to determine your string's width & height)*	*Justify*
Layer	Select destination layer	Layers
Font	Font ID 0..15. Font ID 8..15 provide fastest drawing speed (using BitBlt) and are preferred.	*0..15*
Bcolor	Fill rectangle area defined by X,Y,W,H with this color index. Bcolor 0 = No fill	*0..255*
Fcolor	When using Font ID 8..15 i.e. font ID9, Fcolor will be applied to all pre-existing text that utilize font ID9 When using Font ID 0..7 , Fcolor will only applies to the "this" text being drawn	*0..255*
CLR	1 = Clear the rectangle area before printing any string 0 = Add the string to the rectangle area	*0..1*
Text ID	Instead of an immediate text, a pre-stored (in FLASH) text #nn can be used	*0..95*
Immediate Text	This parameter is optional. If a string is provided, Text ID will be ignored.	-

Example

```
$VL13,300,100,400,500,9,1,2,16,32,1,0,Hello*XX    // Print 'Hello',  L1, Font 2, LR justify in a Rect area(300,100,400,500)
$VL13,300,100, 0,  0, 5,1,2,17,33,1,0,World*XX    // Print 'World',  L1, Font 2, center text @(300,100)
```

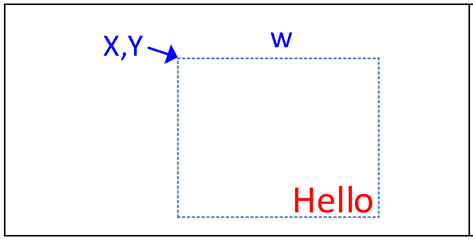


Figure 1

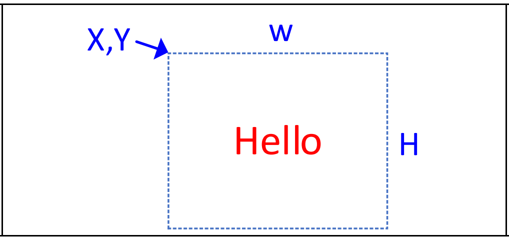


Figure 2

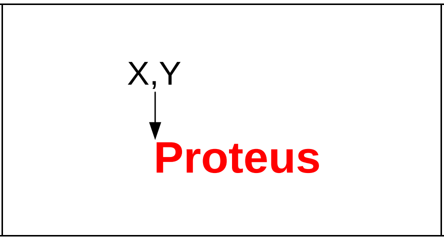


Figure 3

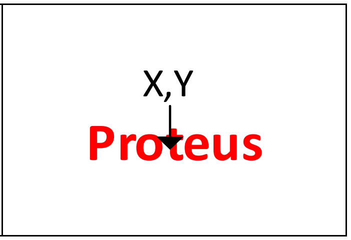


Figure 4

6.5.5 Draw text indirect (Create Text Widget)

Use this command to draw text on the screen indirectly. Text widget is a text field that its content is automatically updated when a known CSV sentence is received. See command $VL29 on how to configure Proteus to receive your unique CSV sentence.

Alternatively, text widgets can also be created via Proteus UI and without using this command. To do this, press F9, select “Config: Load”, select “CSV” and press ↲. Content of these Text widgets can be updated using command $VL43 or when your unique CSV sentence is received. For more detail see “*Display Text Via RS232/Ethernet/USB*” in the User Manual.

Cmnd ID	Payload											
15	ID	1	OnTime	offTime	0	X,Y,W,H	Bcolor	Fcolor	2	Font	Justify	1

Payload Parm	Description	Range
ID	Text Field Widget ID	*40..87*
OnTime, offTime	Text widget ON when (onTime, offTime) = (0,0) Text widget Flashes when (onTime, offTime) = (non-zero, non-zero) Text widget oneshot when (onTime, offTime) = (non-zero, 0)	Unit 100msec.
Area’s X, Y, W, H	X & Y is upper left corner. W & H is the **area** width & height. *W & H must be larger than text’s W & H or widget may not be displayed properly.*	
Bcolor index	Fill rectangle **area** defined by X,Y,W,H with this color. Bcolor 0 = No fill	0..255
Fcolor index	Draw text using this color	*0..255*
Font	Font ID 0..15. Font ID 8..15 provide fastest drawing speed (using BitBlt) and are preferred.	*8..15*
Justify	Text Justification works best if area’s W & H is greater than text’s W & H.	*Justify*

Example

```
// For this example, use color index 64 for Fcolor & 90 for Bcolor.  Set color indexes to Blue
$VL04,64,0xFC00FF00*XX
$VL04,90,0x3C00FF00*XX
// Create widget#40:  Location @XY=600,500, Size @WH=300,200, Bcolor=90, Fcolor=64, Use Font#10 and Center Justify=5
$VL15,40,1,0,0,0,600,500,300,200,90,64,2,10,5,1*XX

// Display widget#40 on screen
$VL18,40,40*XX

// Show “Vac” in widget#40
$VL43,40,Vac*XX

// Remove widget#40 from screen
$VL19,40*XX

// Display widget#40 on screen
$VL18,40,40*XX

// Show “My Name is James Bond” in widget#40
$VL43,40,My Name is James Bond*XX
```

6.5.6 Display Text Widget

Cmnd ID	Payload	
18	Widget ID	Link Register

Payload Parm	Description	Range
Widget ID	Text Field Widget ID	*40..87*
Link Register	Link widget to this register. When content of this register changes, widget is updated automatically.	*3..245*

Example
`$VL18,40,65*XX              // Display widget#40.  Link it to Reg#65.  When Reg#65 changes, widget is updated automatically`

6.5.7 Remove Text Widget

Cmnd ID	Payload	
19	Widget IDn	Widget IDm (optional)

Payload Parm	Description	Range
Widget IDn	First Text Widget ID to remove	*40..87*
Widget IDm	Last Text Widget ID to remove. This field is optional.	*40..87*

Example
`$VL19,40*XX         // Remove widget#40`
`$VL19,40,43*XX      // Remove widget#40,41,42,43`

6.5.8 Define CSV sentence header

Cmnd ID	Payload	
29	Which	header

Parm	Description	Range
Which	'A' for sentence A. 'B' for sentence B. 'C' for sentence C. 'D' for sentence D.	*A,B,C,D*
header	Header	*ASCII*
Example		
`$VL29,A,$PTCF*XX      //Set CSV sentence-A header. Parsed values will be stored in registers #40..51`		
`$VL29,B $MYCSV*XX     //Set CSV sentence-B header. Parsed values will be stored in registers #52..63`		
`$VL29,C,$GNGNS*XX     //Set CSV sentence-C header. Parsed values will be stored in registers #64..75`		
`$VL29,D,$GNGSA*XX     //Set CSV sentence-D header. Parsed values will be stored in registers #76..87`		

See Proteus user manual section 'DISPLAY VALUES FROM ANY CSV SENTENCE' for additional detail on how CSV sentence works.

DRAW

6.5.9 SET DRAWING ATTRIBUTE

Enable/Disable drawing attribute. If enabled, pixels matching ARGB=0 will not be drawn when using display image command $VL25. This option allows superimposing PNG images over another.

Cmnd ID	Payload
08	control

Parm	Description	Range
control	0, Disable, 1 = Enable	*0..1*
Example		
$VL08,0*XX $VL08,1*XX	//Disable Drawing attribute //Enable Drawing attribute	

6.5.10 DOT

Draw one dot with a specified color at a specified location.

Cmnd ID	Payload			
12	Dx	Dy	Layer	Color

Parm	Description	Range
Dx	Destination location X	*0..1920*
Dy	Destination location Y	*0..1080*
Layer	Identifies the layer to draw the dot	Layers
Color	Dot color index	Colors
Example		
$VL12,400,300,0,25*XX	//Draw one dot @400,300 on layer 0. Use color index 25	

6.5.11 LINE

Draw a line from (Sx,Sy) to (Ex,Ey)

Cmnd ID	Payload							
47	Sx	Sy	Ex	Ey	Layer	Size	End	Color

Payloads	Description	Range
Sx	Start X	*0..1920*
Sy	Start Y	*0..1080*
Ex	End X	*0..1920*
Ey	End Y	*0..1080*
Layer	Layer	Layers
Size	Line size	*0..15*
End	End of line (round, flat)	*0..1*
Color	Line color index	0..255

Example

```
$VL47,100,100,200,200,1,1,1,31*XX    // Draw Line from (100,100) to (200,200), layer 1, 1 pixel thick, color index 31
$VL47,200,200,600,300,0,2,1,32*XX    // Draw Line from (10,  25) to (600,300), layer 0, 2 pixel thick, color index 32
```

6.5.12 TRIANGLE

Cmnd ID	Payload								
52	X1	Y1	X2	Y2	X3	Y3	color	Layer	Fill	Size

Payloads	Description	Range
X1,Y1	Vertex 1	*X = 0..1920, Y = 0..1080*
X2,Y2	Vertex 2	*X = 0..1920, Y = 0..1080*
X3,Y3	Vertex 3	*X = 0..1920, Y = 0..1080*
Color	Line color	0..255
Layer	Layer	Layers
Fill	0 = no fill, 1 = fill	*0,1*
Size	Size of the line if triangle is not filled	*0..15*
Example		
`$VL52,100,100,300,50,300,300,26,1,1,0*XX // Draw Triangle vertex @(100,100),(200,200),(300,300), L0, fill`		

6.5.13 SQUARE

Cmnd ID	Payload							
55	X	Y	W	H	color	Layer	0	Size

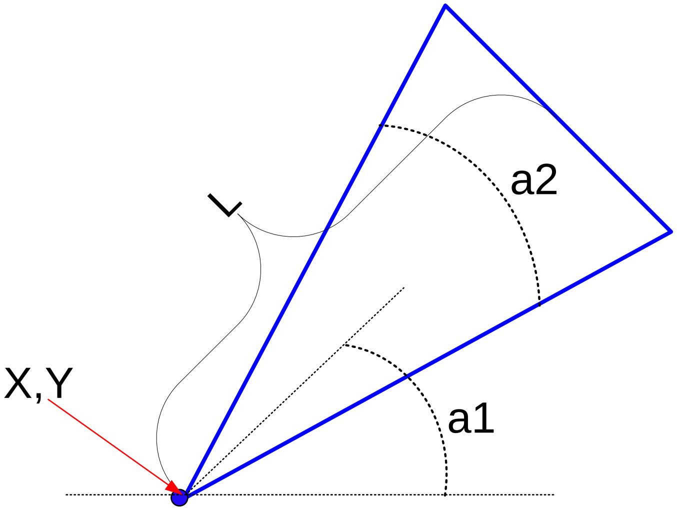
Payloads	Description	Range
X		*0..1920*
Y		*0..1080*
W		*0..1920*
H		*0..1080*
Color	Line color index	0..255
Layer	Layer	Layers
0	Unused	*0*
Size	Size of the line if square is not filled	*0..15*
Example		
`$VL55,100,100,1000,300,26,1,0,15*XX`		

6.5.14 Cone

Draw a vector (line with an arrow)

Cmnd ID		Payload					
54	X	Y	L	a1	a2	Color	Layer

Payloads	Description	Range
X, Y	See drawing	*0..1920*
L	Length of the cone. See drawing	0..1920
a1	Angle of the cone in degree. See drawing	*0..360*
a2	Arc of the cone in degree. See drawing	*0..180*
Color	Line color index	0..255
Layer	Layer	Layers
Example		
$VL54,300,300,100,45,25,29,1*XX // Draw cone at (300,300), length 100, angle 45, arc 25,L1		



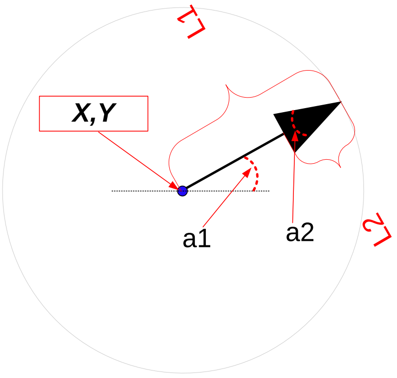
6.5.15 Vector

Draw a vector (line with an arrow).

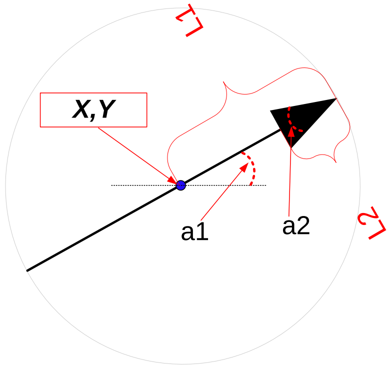
Cmnd ID	Payload									
53	X	Y	L1	a1	L2	a2	Color	Layer	Size	Type

Payloads	Description	Range
X,Y	See drawing	*0..1920*
L1	Length of the vector. See drawing	0..1920
a1	Angle of the vector in degree. See drawing	*0.360*
L2	Length of the arrow. See drawing	0..1920
a2	Arc of the arrow in degree. See drawing	*0..90*
Color	Line color	0..255
Layer	Layer	Layers
Size	Width of the line in pixels	*0..9*
Type	See drawing below	0..1
Example		
$VL53,300,300,100,45,20,20,30,1,1,0*XX	// Draw Vector at (300,300),length 100, angle 45, L0	

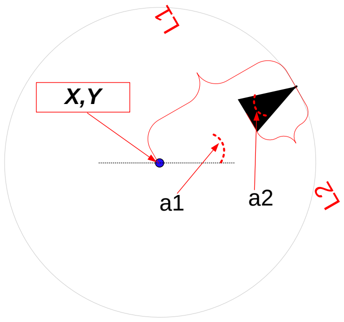
Type 0



Type 1



Type 2



6.5.16 Grid Lines

Draw grid lines on layer 0.

Cmnd ID	Payload			
48	X pixels/division	Y pixels/division	layer	color

Parm	Description	Range
X pixels/division	Number of pixels between vertical grid lines	0..1920
Y pixels/division	Number of pixels between horizontal grid lines	0..1080
layer	Layer	Layers
color	Color index	0..255

Example

```
$VL48,50,50,3,35*XX                      // X pixel/div =50, Y pixel/div = 50
$VL48,100,100,0,35*XX                    // X pixel/div =25, Y pixel/div =100
```

6.6 Macro

6.6.1 Save

A macro is a series of commands that are executed one after the other in the same order. They're very practical to automate repetitive tasks. This command allows for downloading a macro into FLASH.

Cmnd ID	Payload	
20	Macro ID	CommandStream[]

Payloads	Description	Range
Macro ID	Two-digit ID e.g., 00,01,09,25	00..31
CommandStream[]	Group of commands	ASCII

The **CommandStream**[] array contains commands to be executed. Each command consists of Cmnd ID + Payload followed by '@' character.

6.6.1.1 Example 1:

We would like to assign the following 3 commands to Macro 10:

Commands	*Arguments*
Command 1	`$VL13,960,540,400,500,7,1,2,16,1,0,65*XX`
Command 2	`$VL13,960,540,400,500,5,1,5,16,0,0,455*XX`
Command 3	`$VL13,960,540,400,500,3,1,2,16,0,0,720+*XX`

The command 'Macro Save' would be constructed as follow:

```
$VL20,10,$VL13,960,540,400,500,9,1,2,16,1,0,65@$VL13,960,540,400,500,9,1,2,16,1,0,455@$VL13,960,540,400,500,9,1,2,16,1,0,720+@*XX
```

To run Macro 10, issue Macro Execute command **$VL21,10***XX

6.6.1.2 Example 2:

We would like to assign the same commands to Macro 11. However, when executing this macro, we would like to pass parameters x, y and strings as arguments. To accomplish this, parameters (x, y, string x 3) should be replaced with argument index '^#'.

Commands	*Arguments*
Command 1	`$VL13,^0,^1,400,500,7,1,2,16,1,0,^2*XX`
Command 2	`$VL13,^0,^1,400,500,5,1,5,16,0,0,^3*XX`
Command 3	`$VL13,^0,^1,400,500,3,1,2,16,0,0,^4*XX`

The command 'Macro Save' would be constructed as follow:

```
$VL20,11,$VL13,^0,^1,400,500,9,1,2,16,1,0,^2@$VL13,^0,^1,400,500,9,1,2,16,0,0,^3@$VL13,^0,^1,400,500,9,1,2,16,0,0,^4@*XX
```

To run Macro 11, issue Macro Execute command **$VL21,11,960,540,65,455,720+***XX

6.6.2 Execute

Execute a pre-defined Macro.

Cmnd ID	Payload	
21	Macro ID	Arguments (optional)

Payloads	Description	Range
Macro ID	ID	*0..31*
Arguments (optional)	Required if Macro expects data	*ASCII*
Example		
$VL21,22*XX	// Execute Macro #22. No argument	
$VL21,23,Hello World*XX	// Execute Macro #23 with argument 'Hello World'	

6.6.3 Delete

Delete one or a series of sequential Macros from FLASH.

Cmnd ID	Payload	
22	IDm	IDn

Payloads	Description	Range
IDm	First macro to be deleted	*0..31*
IDn	Last macro to be deleted	*0..31*
Example		
$VL22,10,10*XX	// Delete Macro 10	
$VL22,10,12*XX	// Delete Macro 10,11,12	

6.6.4 Get CRC

Request CRC of each Macro.

Cmnd ID	Payload	
24	IDm	IDn (optional)

Payloads	Description	Range
IDm	*First Macro to receive CRC's from*	*0..31*
IDn (optional)	*Last Macro to receive CRC's from*	*0..31*
Example		
$VL24,10*XX	// Report CRC of Macros 10	
Reply:		
<STX>24,512ACB65,XX<ETX>		

6.7 IMAGE

6.7.1 DISPLAY

Display the entire image or part of an image at a specific location.

Cmnd ID	Payload		
25	Image ID	Dx	Dy

Payloads	Description	Range
Image ID	ID	*0..95*
Dx	Destination location X	*0..1920*
Dy	Destination location Y	*0..1080*
Example		
`$VL25,10,100,100*XX`	`// Draw Image #10 @(100,100)`	
`$VL25,14,500,500*XX`	`// Draw Image #14 @(500,500)`	

6.8 Get Status

Cmnd ID	Payload
35	0
Example	
$VL35,0*XX	

Reply will be sent as shown below:

Reply ID	Payload			
35	Video Format	watchdog	Cold Boot	Version

Parm	Description	Range
Video Format	Video VIC code i.e. 4, 5, 6, 19, 20, 21, 32, 33, 34	-
Watchdog	0 = Watchdog is disable 1 = Watchdog is enabled	0..1
Cold Boot	0 = Proteus did not lose power 1 = Proteus lost power	0..1
Version	Firmware version	*String*
Example		
<STX>35,21,0,1,V1.2,CS<ETX> // Video 1080p, watchdog is disabled, Proteus lost power since last status request		

6.9 Registers

Proteus system contains a collection of registers used for configuring the system and accessing the data it produces. These registers may be read or written to using the Read Register and Write Register commands (refer to SCS for detail). The table below provides a quick reference for all of the registers and their associated properties. The device specific (Cineflex, IMU, GPS and etc) registers are automatically updated when the associated device is connected to Proteus. Widgets that are associated to a register are updated automatically when the content of the register changes.

Refer to appendix G of the user manual for complete list of register designation.

6.9.1 Set Register

Set a register (range 3..255) to a desire value. Any Widgets linked to the register will be automatically updated.

Cmnd ID	Payload	
43	Register ID	Value

Parm	Description	Range
Register ID	ID of the register being set	*3..255*
Value	Register value	-

Example

```
$VL43,27,+30,645327*XX                   // Set GPS Latitude to,+30,645327
$VL43,30,-45.35*XX                       // Set Navigation Pitch to -45.35 degree
$VL43,81,1,22,333,4444,55555*XX          // Set registers 81=1, 82=22, 83=333, 84=4444, 85=55555
```

6.9.2 Get Register

Cmnd ID	Payload
42	Register ID

Parm	Description	Range
Register ID	ID of the register to query	*3..255*

Command:

```
$VL42,27*XX                              // Get Latitude
```

Reply:

```
<STX>42,N33 38.4722,CS<ETX>              // Latitude
```

Real Time Clock

6.9.3 Set Time

Cmnd ID	Payload
44	Time

Parm	Description	Range
Time	HH:MM:SS	*00:00:00 - 23:59:59*
Example		
$VL44,08:30:15*XX	// Set time to 8:30:15	

6.9.4 Set Date

Cmnd ID	Payload
45	Date

Parm	Description	Range
Date	MM/DD/YY	*10/15/12*
Example		
$VL45,10/15/19*XX	// Set date to Oct 15, 2012	

6.9.5 Set GPS Time Zone

Cmnd ID	Payload
46	Time Offset

Parm	Description	Range
Time Offset	-HH:MM or +HH:MM	*00:00 .. 23:59*
Example		
$VL46,-15:30*XX	// Offset UTC by -15:30	
$VL46,+11:30*XX	// Offset UTC by +11:30	

6.10 Display/Remove Widgets

This command allows users to remotely enable/disable any text widget (time, date, csv token, analog, quadrature, etc.) or app (ROV & Plane Situation Awareness, Reticle, XY measurements, sliders, etc.)

Cmnd ID	Payload	
16	Register ID	Control

Parm	Description	Range
Register ID	Refer to Appendix A for register ID	*1..255*
Control	0 = Remove 1 = Display	*0-1*

Example

```
$VL16,52,1*XX                           // Display text Widget #52 (Token B1) on video screen
$VL16,52,0*XX                           // Remove text Widget #52 (Token B1) from video screen
$VL16,238,1*XX                          // Display Reticle App
$VL16,250,1*XX                          // Display Slider 1.
$VL16,247,1*XX                          // Display XY Measurement App
$VL16,165,1*XX                          // Display User text #1
$VL16,7,1*XX                            // Display IRIG Time
$VL16,93,1*XX                           // Display GPS Latitude
$VL16,25,1*XX                           // Display Quadrature Map Count#1
```

Prior to using command 16, use the built-in GUI as described in the User Manual ("Insert Variable from CSV sentence" and "Insert Text") to position the desire tokens or user texts on the screen with the desire font, foreground color, background color, text justification, window width, etc.

6.11 Flash Text Widgets

This command allows users to display/remove/flash any text widget

Cmnd ID	Payload	
57	Text Widget ID	Control

Parm	Description	Range
Text Widget ID	Refer to Appendix A for Text Widget ID	*40..87*
Control	0 = Remove 1 = Display 2 = Flash 3 = One shot	*0-4*

Example

```
$VL57,52,1*XX                           // Display text Widget #52 (Token B1)
$VL57,52,0*XX                           // Remove  text Widget #52 (Token B1)
$VL57,52,2*XX                           // Flash   text Widget #52 (Token B1) 100ms on, 100msec off
$VL57,52,3*XX                           // Display Text Widget #52 (Token B1) for 400msec and then remove
```

6.12 Configure Quadrature Counter

Cmnd ID	Payload			
27	Channel	Slope	Intercept	Reset

Parm	Description	Range
Channel	0=Counter #1, 1=Counter#2	*0..1*
Slope	Any floating value i.e., 1.005674	-
Intercept	Any floating value i.e., 2.674765	*3..255*
Reset	1= Reset counter	-

Example

```
$VL27,0,1,0,1*XX                          // Ch=0, Slope=1, Intercept=0, Reset counter
$VL27,1,1.005674,2.674765,0*XX            // CH=1, Slope=1.005674, Intercept=2.674765, Do not reset counter
```

6.13 Emulate keyboard

This command allows the user to emulate keyboard through RS232.

Cmnd ID	Payload			
56	Key code	Ctrl-left	Alt-left	Shift-left

Parm	Description	Range
Key code	Scan code	0.FF
Ctrl-left	Ctrl	0..1
Alt-left	Alt	0..1
Shift-left	Shift	0..1

Example

```
$VL56,BB,0,0,0*XX                  // Emulate F1
$VL56,BC,0,0,0*XX                  // Emulate F2
$VL56,37,0,0,0*XX                  // Emulate 7
$VL56,C4,0,0,0*XX                  // Emulate F10
$VL56,1B,0,0,0*XX                  // Emulate ESC
$VL56,BB,1,0,0*XX                  // ALT_LEFT + F1
$VL56,BB,0,1,0*XX                  // CTRL_LEFT + F1
$VL56,BB,1,1,0*XX                  // CTRL_LEFT + ALT_LEFT + F1
$VL56,BB,1,1,1*XX                  // CTRL_LEFT + ALT_LEFT + SHIFT_LEFT + F1
```

In order to obtain the “key code” for any other key, display “*Development*” register as described below and press your desire key. The display format will be `Keycode + Ctrl-left:Alt-left:Shift-left`

- Press F9

- Select “Display: Device data”
- Select “Miscellaneous”

6.14 Refresh Screen

Refresh screen as if ESC key were pressed.

Cmnd ID	Payload
17	-

Parm	Description	Range
Example		
$VL17*XX		

6.15 Reset

This command allows the user to reset Proteus through RS232 command

Cmnd ID	Payload
36	RESET PROTEUS

Parm	Description	Range
RESET PROTEUS		-
Example		
$VL36,RESET PROTEUS *XX		

Appendix A – Register Designation

Register ID	Designation
1	
2	
3	PM_HDMI_FORMAT,
4	PM_UTC_OFFSET,
5	PM_RTC_TIME,
6	PM_RTC_DATE,
7	PM_IRIG_TIME,
8	PM_IRIG_DATE,
9	PM_ATC_TIME,
10	
11	PM_NTP_BUFFER,
12	PM_UNIX_EPOCH,
13	
14	PM_UP_TIMER,
15	PM_AN_RAW1,
16	PM_AN_RAW2,
17	PM_AN_RAW3,
18	PM_AN_RAW4,
19	PM_AN_MAP1,
20	PM_AN_MAP2,
21	PM_AN_MAP3,
22	PM_AN_MAP4,
23	PM_QUAD_RAW1,
24	PM_QUAD_RAW2,
25	PM_QUAD_MAP1,
26	PM_QUAD_MAP2,
27	PM_COUNTER1,

28	PM_COUNTER2,
29	PM_LSR,
30	PM_IP_ADDRESS,
31	PM_GPI,
32	PM_DEVELOPMENT,
33	PM_FORMAT,
34	PM_SDI_FORMAT,
35	PM_TARGET_LAT,
36	PM_TARGET_LON,
37	PM_PROTEUS_HEADING,
38	PM_PROTEUS_PITCH,
39	PM_PROTEUS_ROLL,
40	PM_TOKENA1,
41	PM_TOKENA2,
42	PM_TOKENA3,
43	PM_TOKENA4,
44	PM_TOKENA5,
45	PM_TOKENA6,
46	PM_TOKENA7,
47	PM_TOKENA8,
48	PM_TOKENA9,
49	PM_TOKENA10,
50	PM_TOKENA11,
51	PM_TOKENA12,
52	PM_TOKENB1,
53	PM_TOKENB2,
54	PM_TOKENB3,
55	PM_TOKENB4,
56	PM_TOKENB5,

57	PM_TOKENB6,
58	PM_TOKENB7,
59	PM_TOKENB8,
60	PM_TOKENB9,
61	PM_TOKENB10,
62	PM_TOKENB11,
63	PM_TOKENB12,
64	PM_TOKENC1,
65	PM_TOKENC2,
66	PM_TOKENC3,
67	PM_TOKENC4,
68	PM_TOKENC5,
69	PM_TOKENC6,
70	PM_TOKENC7,
71	PM_TOKENC8,
72	PM_TOKENC9,
73	PM_TOKENC10,
74	PM_TOKENC11,
75	PM_TOKENC12,
76	PM_TOKEND1,
77	PM_TOKEND2,
78	PM_TOKEND3,
79	PM_TOKEND4,
80	PM_TOKEND5,
81	PM_TOKEND6,
82	PM_TOKEND7,
83	PM_TOKEND8,
84	PM_TOKEND9,
85	PM_TOKEND10,

86	PM_TOKEND11,
87	PM_TOKEND12,
88	PM_GPS_ALTITUDE,
89	PM_GPS_COG,
90	PM_GPS_SPEED,
91	PM_GPS_TIME,
92	PM_GPS_DATE,
93	PM_GPS_LAT_D,
94	PM_GPS_LON_D,
95	PM_GPS_LAT_DM,
96	PM_GPS_LON_DM,
97	PM_GPS_LAT_DMS,
98	PM_GPS_LON_DMS,
99	PM_GPS_SEQUENCE,
100	PM_GPS_ID,
101	PM_GPS2_ALTITUDE,
102	PM_GPS2_COG,
103	PM_GPS2_SPEED,
104	PM_GPS2_TIME,
105	PM_GPS2_DATE,
106	PM_GPS2_LAT_D,
107	PM_GPS2_LON_D,
108	PM_GPS2_LAT_DM,
109	PM_GPS2_LON_DM,
110	PM_GPS2_LAT_DMS,
111	PM_GPS2_LON_DMS,
112	PM_GPS2_SEQUENCE,
113	PM_GPS2_ID,
114	PM_IMU_HEADING,

115	PM_IMU_PITCH,
116	PM_IMU_ROLL,
117	PM_IMU_HEIGHT,
118	PM_IMU_LAT,
119	PM_IMU_LON,
120	PM_IMU_TIME,
121	PM_IMU_DATE,
122	PM_IMU_LAT_DMS,
123	PM_IMU_LON_DMS,
124	PM_CINEFLEX_AZIMUTH,
125	PM_CINEFLEX_ELEVATION,
126	PM_CINEFLEX_ROLL,
127	PM_CINEFLEX_FOCUS,
128	PM_CINEFLEX_ZOOM,
129	PM_CINEFLEX_IRIS,
130	PM_CINEFLEX_TELE,
131	PM_CINEFLEX_PAN,
132	PM_ALTIMETER,
133	PM_VSPEED,
134	PM_MWV_ANGLE,
135	PM_MWV_REFERENCE,
136	PM_MWV_SPEED,
137	PM_MWV_UNIT,
138	PM_DBT_DEPTH,
139	PM_DPT_DEPTH,
140	PM_DPT_OFFSET,
141	PM_DPT_RANGE,
142	PM_MTW_TEMPRATURE,
143	PM_LAT_FORGGARMC,

144	PM_LON_FORGGARMC,
145	
146	
147	PM_HEADING,
148	PM_PITCH,
149	PM_ROLL,
150	PM_HCC_HEADING,
151	PM_DBS_DEPTH,
152	PM_PCIT_TILT,
153	PM_PCIPR_PITCH,
154	PM_PCIPR_ROLL,
155	WIG_RETICLE_X,
156	WIG_RETICLE_Y,
157	
158	
159	
160	
161	WIG_MARKER_DX_RAW,
162	WIG_MARKER_DY_RAW,
163	WIG_MARKER_DX_MAP,
164	WIG_MARKER_DY_MAP,
165	PM_TEXT1,
166	PM_TEXT2,
167	PM_TEXT3,
168	PM_TEXT4,
169	PM_TEXT5,
170	PM_TEXT6,
171	PM_TEXT7,
172	PM_TEXT8,

173	PM_TEXT9,
174	PM_TEXT10,
175	PM_TEXT11,
176	PM_TEXT12,
177	PM_TEXT13,
178	PM_TEXT14,
179	PM_TEXT15,
180	PM_TEXT16,
181	PM_TEXT17,
182	PM_TEXT18,
183	PM_TEXT19,
184	PM_TEXT20,
185	PM_TEXT21,
186	PM_TEXT22,
187	PM_TEXT23,
188	PM_TEXT24,
189	PM_TEXT25,
190	PM_TEXT26,
191	PM_TEXT27,
192	PM_TEXT28,
193	PM_TEXT29,
194	PM_TEXT30,
195	PM_IMAGE1,
196	PM_IMAGE2,
197	PM_IMAGE3,
198	PM_IMAGE4,
199	PM_IMAGE5,
200	PM_IMAGE6,
201	PM_IMAGE7,

202	PM_IMAGE8,
203	PM_IMAGE9,
204	PM_IMAGE10,
205	PM_IMAGE11,
206	PM_IMAGE12,
207	PM_IMAGE13,
208	PM_IMAGE14,
209	PM_IMAGE15,
210	PM_IMAGE16,
211	PM_IMAGE17,
212	PM_IMAGE18,
213	PM_IMAGE19,
214	PM_IMAGE20,
215	PM_IMAGE21,
216	PM_IMAGE22,
217	PM_IMAGE23,
218	PM_IMAGE24,
219	PM_IMAGE25,
220	PM_IMAGE26,
221	PM_IMAGE27,
222	PM_IMAGE28,
223	PM_IMAGE29,
224	PM_IMAGE30,
225	PM_PGN_TIME,
226	PM_PGN_DATE,
227	PM_PGN_LAT,
228	PM_PGN_LON,
229	PM_PGN_SPEED_WATER_REF,
230	PM_PGN_SPEED_GROUND_REF,

231	PM_PGN_SENSOR_DEPTH,
232	PM_PGN_WIND_SPEED,
233	PM_PGN_WIND_DIR,
234	PM_PGN_WIND_GUSTS,
235	PM_PGN_TEMP_AIR,
236	PM_PGN_PRESSURE,
237	PM_PGN_HUMIDITY,
238	WG_RETICLE_ID,
239	PM_POS_SCANNER,
240	
241	
242	
243	
244	
245	
246	
247	WG_MARKER_ID,
248	WG_RCOMPASS_ID,
249	WG_COMPASS_ID,
250	WG_SLIDER0_ID,
251	WG_SLIDER1_ID,
252	WG_SLIDER2_ID,
253	WG_SLIDER3_ID,
254	WG_ROV_ID,
255	WG_PLANE_ID