Proteus-V SCS
Software Communication Specification

Version V1.2
Aug 16, 2019

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            // Command ID=07,    checksum included (1D)
$VL07,400,200,300,50,0,33*XX            // 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<CR><LF>

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

Most commands do not require a formatted reply. In such instances, Proteus transmits a single byte (i.e. ACK or NAK) to indicate command acceptance or rejection status.

Reply character	Description
Binary 6 (ACK)	Message was accepted
Binary 5	Message has incorrect checksum
Binary 9	Invalid Command ID was received
Binary 0 (NAK)	Message was rejected
Binary B	Missing comma delimiter

When a command requires a formatted reply, the reply structure is as shown below:

Start	Command ID	,	Payload	*	Checksum	End
<STX>	ii	,	Reply Parameters	,	cc	<ETX>
02	07	,	0400,0200,0300,0050,0,4070	,	78	03

A complete formatted reply including the STX, ETX and checksum is shown below:

0	3	3	2	3	3	3	3	2	3	3	3	3	2	3	3	3	3	2	3	3	3	5	3	2	3	2	3	3	3	7	3	8	0
2	0	7	C	0	4	0	0	C	0	2	0	0	C	0	3	0	0	C	0	0	5	0	C	0	C	4	0	7	0	C	7	8	3
2	0	7	,	0	4	0	0	,	0	2	0	0	,	0	3	0	0	,	0	0	5	0	,	0	,	4	0	7	0	,	7	8	3

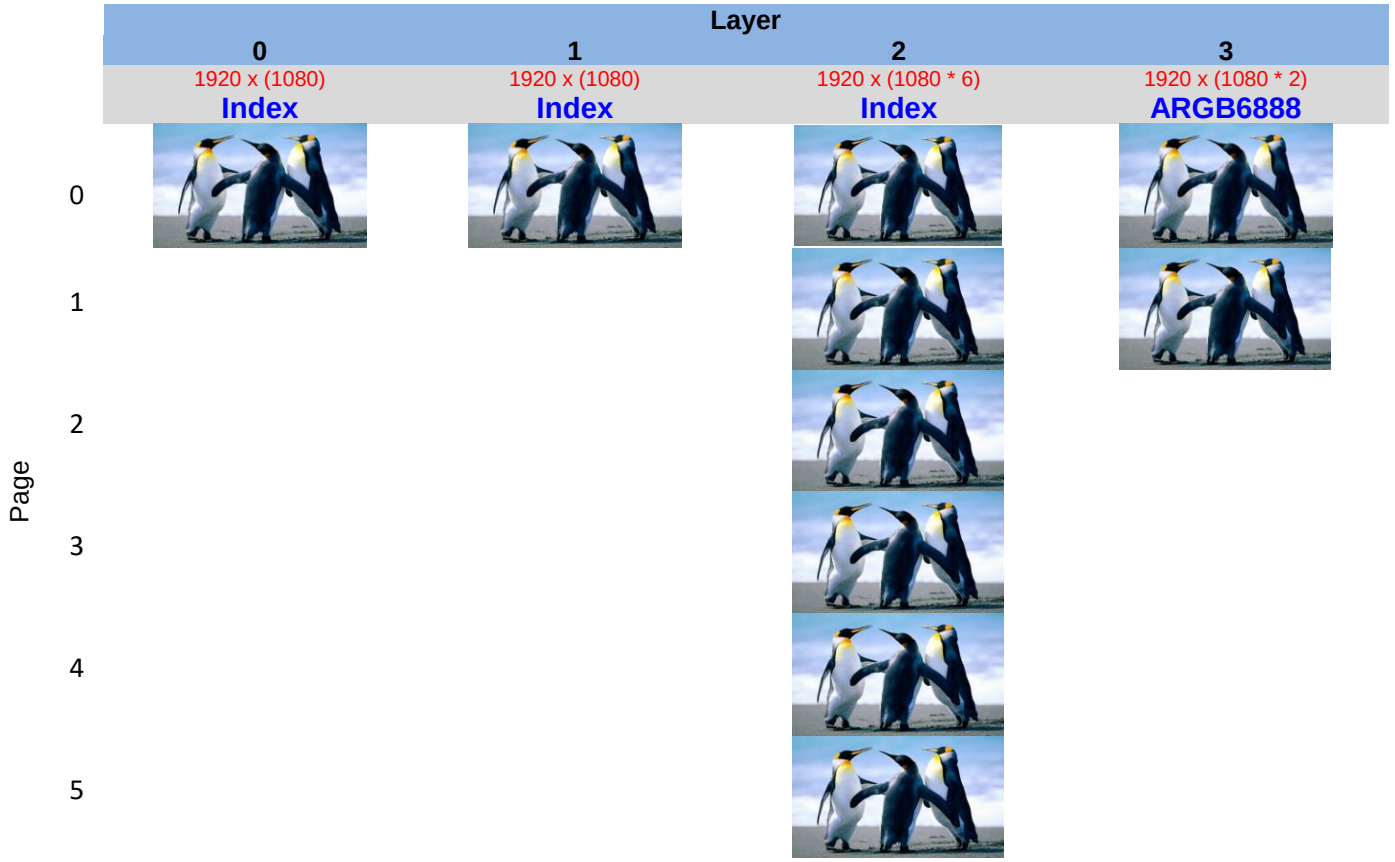
2. 6

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 none-visible portion of the layer 3. Proteus copy commands uses BitBlt to copy objects from the virtual screens to the visible screen at a very high speed.



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	
05	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		
$VL05,2,4*XX	// Set Layer 2 to highest priority	

6.1.3 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

6.1.4 SELECT PAGE

Make alternative page 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	

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	

Font set 0..7 are identical to Font set 8..15.
As shown below, font set 0..7 can use any *color index*. However, font set 8..15 use specific color index.

Font ID	Font Type	Font Color-Index
0	FLASH	1..255 *(default 48)*
1	FLASH	1..255 *(default 49)*
2	FLASH	1..255 *(default 50)*
3	FLASH	1..255 *(default 51)*
4	FLASH	1..255 *(default 52)*
5	FLASH	1..255 *(default 53)*
6	FLASH	1..255 *(default 54)*
7	FLASH	1..255 *(default 55)*
8	VRAM	56
9	VRAM	57
10	VRAM	58
11	VRAM	59
12	VRAM	60
13	VRAM	61
14	VRAM	62
15	VRAM	63

To change Font#11 color to white, issue the following command: `$VL04,59,FCFFFFFF*XX`
Font set 8..15 provide fastest drawing speed (using BitBlt) and are preferred.

6.3 RECTANGLE

6.3.1 COPY

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		

6.3.2 PAINT OR ERASE

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		

6.4 STRING

6.4.1 STORE

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	

6.4.2 DELETE

Delete one or more strings from FLASH.

Cmnd ID	Payload	
70	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		
$VL70,25,25*XX	// Delete string# 25	
$VL70,25,29*XX	// Delete string# 25,26,27,28,29	

6.4.3 DRAW

Draw and justify string 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	String ID	String (optional)	

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

Payload Parm	Description	Range
X	A rectangle area in which to justify the string. Rectangle area can be filled with color index Bcolor	*0..1920*
Y		*0..1080*
W		*0..1920*
H		*0..1080*
Justify	Select justification	*Justify*
Layer	Select destination layer	Layers
Font	When using Font set 8..15 i.e. FON10, Fcolor will be applied to any currently visible text that uses FONT10 When using Font set 0..7 , Fcolor will only apply to the text being drawn. Font set 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	Font Color	*0..255*
CLR	1 = Clear the rectangle area before printing any string 0 = Add the string to the rectangle area	*0..1*
String nn	Instead of an immediate string, a pre-stored (in FLASH) string #nn can be used	*0..95*
Immediate String	This parameter is optional If a string is provided, String #nn 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)
$VL13,300,100, 0,  0, 0,1,2,18,34,1,0,Proteus*XX  // Print 'Proteus',       L1, Font 2, start printing text @(300,100)
```

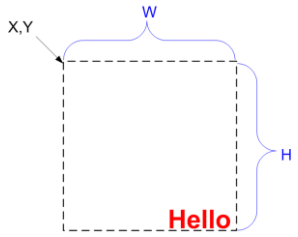


Figure 1

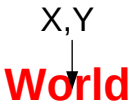


Figure 2

Figure 3

6.4.4 WIDGET

Draw and justify strings on top of an image exclusively on layer 3.

Cmnd ID	Payload											
14	X	Y	W	H	Justify	3	Font	Image	CLR	String ID	String (optional)	

Examples:

14	X	Y	W	H	7	3	4	2	1	0	Left	LL justify text in Rect area X,Y,W,H. Erase area 1st
14	X	Y	H	H	5	3	4	2	0	0	Center	CC justify text in Rect area X,Y,W,H.
14	X	Y	W	H	3	3	4	2	0	0	Right	UR justify text in Rect area X,Y,W,H

Payload Parm	Description	Range
X	• Widget's upper left corner will be positioned at X & Y. • User provided W, H will be replaced with image W &H. However, if the image does not exist, the user provided W & H will be used.	*0..1920*
Y		*0..1080*
W		*0..1920*
H		*0..1080*
Justify	Justification code	*Justify*
Font	Select a Font	*0..15*
Image	Image to be displayed as background	*0..95*
CLR	1 = Clear rectangle area (W & H) before printing string, 0 = Add text to the rectangle area	*0..1*
String ID	Pre-stored string ID.	*0..95*
String (optional)	Immediate string. If this string is provided, String ID is ignored	-

Example

```
$VL14,300,100,169,234,7,3,10,1,1,0,Left*XX
$VL14,300,100,169,234,5,3,4,2,0,0,Center*XX
$VL14,300,100,169,234,3,3,4,2,0,0,Right*XX
```

Figure 2

6.5 Draw

6.5.1 Dot

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

Cmnd ID	Payload			
02	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		
$VL02,400,300,0,25*XX	//Draw one dot @400,300 on layer 0. Use color index 25	

6.5.2 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	

6.5.3 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	1 = fill, 0=Do not 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`		

6.5.4 SQUARE

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

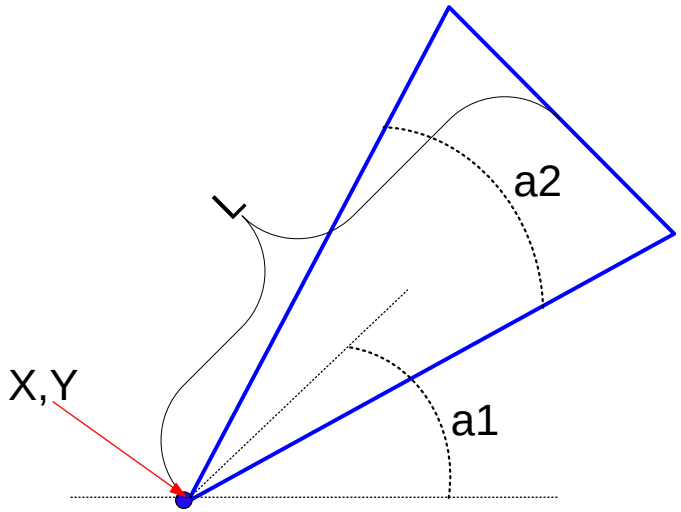
Payloads	Description	Range
X		*0..1920*
Y		*0..1080*
W		*0..1920*
H		*0..1080*
Color	Line color index	0..255
Layer	Layer	Layers
0	Unused	*0*
Size	Size of the line if square is not filled	*0..15*
Example		
`$VL55,100,100,1000,300,26,1,1,15*XX`		

6.5.5 CONE

Draw a vector (line with an arrow)

Cmnd ID		Payload					
4	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		



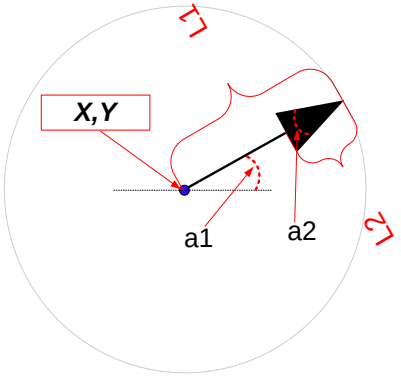
6.5.6 VECTOR

Draw a vector (line with an arrow).

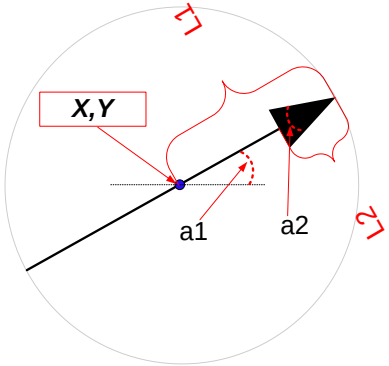
Cmnd ID	Payload									
53	X	Y	L1	a1	L2	a2	Color	Layer	Size	Type

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`	

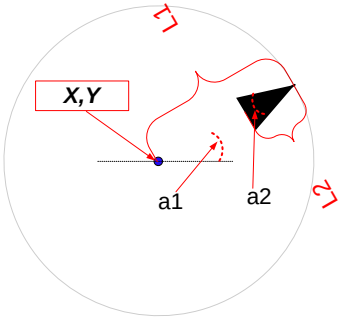
Type 0



Type 1



Type 2



6.5.7 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	Assigned ID	0..95
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..95*
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..95*
IDn	Last macro to be deleted	*0..95*
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..95*
IDn (optional)	*Last Macro to receive CRC's from*	*0..95*
Example		
$VL24,10*XX	// Report CRC of Macros 10	
$VL24,10,18*XX	// Report CRC of Macros 10..18	

6.6.5 DISPLAY

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

Cmnd ID	Payload			
25	Image ID	Dx	Dy	layer

Payloads	Description	Range
Image ID	ID	*0..95*
Dx	Destination location X	*0..1920*
Dy	Destination location Y	*0..1080*
Layer	3	*0*
Example		
$VL25,10,100,100,3*XX	// Draw Image #10 @(100,100) on Layer 3	
$VL25,14,500,500,3*XX	// Draw Image #14 @(500,500) on Layer 3	

6.7 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		

6.8 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.

6.8.1 DISPLAY REGISTER

Cmnd ID	Payload	
57	Register ID	Display mode

Parm	Description	Range
Register ID	ID of the register being displayed	*40..51* *(Token A1..A12)* *52..63* *(Token B1..B12)* *64..75* *(Token C1..C12)* *76..87* *(Token D1..D12)* *165..194* *(User Text 1..30)*
Display mode	0 = Remove from display 1 = Display 2 = Flash (1s ON, 1s OFF) 3 = One shot (1s ON and then remove)	*0-3*

Example

```
$VL57,52,0*XX                    // Remove  text Widget #52 (Token A1) from the video screen
$VL57,52,1*XX                    // Display text Widget #52 (Token A1) on the video screen
$VL57,52,2*XX                    // Flash   text Widget #52 (Token A1) at 1Hz rate
$VL57,52,3*XX                    // Display text Widget #52 (Token A1) for 1second and then remove.
$VL43,52,Hello World*XX          // Display 'Hello World' in the text Widget #52
$VL43,52,Jan 1,2018*XX           // Display 'Jan 1,2018' in the text Widget #52
```

Prior to using command 57, 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 and select the desire font, foreground color, background color, text justification, window width, etc.

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	

6.8.2 GET REGISTER

Cmnd ID	Payload
42	Register ID

Parm	Description	Range
Register ID	ID of the register to query	*3..255*
Example		
$VL42,27*XX	// Get Latitude	

Reply will be as follow:

Reply ID	Payload
42	Register data

Parm	Description	Range
Register data	Data	-
Reply Example:		
<STX>42,N33 38.4722,CS<ETX>	// GPS Latitude	

REAL TIME CLOCK

6.8.3 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	

6.8.4 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	

6.8.5 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	

6.9 EMULATE KEYBOARD

This command allows the user to emulate keyboard through RS232 commands

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

Parm	Description	Range
Key code	Scan code	0.FF
Key state	Alt	0..1
Key state	Ctrl	0..1

Example

```
$VL56,BB,0,0*XX                 // Emulate F1
$VL56,BC,0,0*XX                 // Emulate F2
$VL56,BD,0,0*XX                 // Emulate F3
$VL56,7, 0,0*XX                 // Emulate 7
$VL56,C3,0,0*XX                 // Emulate F9
$VL56,C4,0,0*XX                 // Emulate F10
$VL56,1B,0,0*XX                 // Emulate ESC
$VL56,BB,1,0*XX                 // ALT_LEFT + F1
$VL56,BB,0,1*XX                 // CTRL_LEFT + F1
$VL56,BB,1,1*XX                 // CTRL_LEFT + ALT_LEFT + F1
```