Proteus-IV SCS
Software Communication Specification

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=1D
$VL07,400,200,300,50,0,33*XX            // Command ID=07, ignore checksum
```

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 computation can be bypassed 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	2	3	3	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		

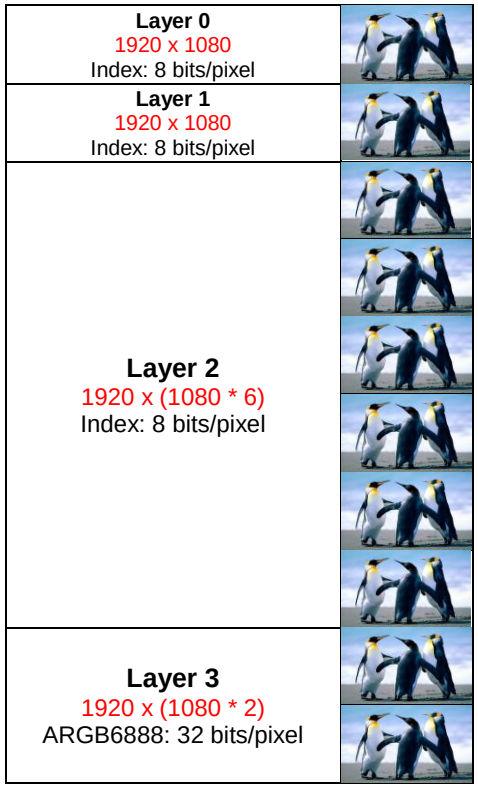
2. 6

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. The additional frame pages in layer 2 and 3 are not visible on the monitor and is commonly referred to as virtual screen.

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 CLEAR SCREEN

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*	

6.2 SET ALPHA BLEND

Cmnd ID	Payload			
03	Blend 0	Blend 1	Blend 2	Blend 3

Parm	Description	Range
Blend 0	Blend applied to layer 0	*0..63*
Blend 1	Blend applied to layer 1	*0..63*
Blend 2	Blend applied to layer 2	*0..63*
Blend 3	Blend applied to layer 3	*0..63*
Example		
$VL03,20,30,40,50*XX		

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

6.3 COLOR

6.3.1 INDEX

Cmnd ID	Payload	
04	Index	Color

Parm	Description	Range
Index	Color Index	0..255
Color	aaRRGGBB where aa is alpha blend. For aa, use only most significant 6-bits i.e. 00..FC (aaaaaa00)	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	

As shown below, each font has an assigned color.

Font ID	Font Color-Index	Text-Widget Background Color-Index
0	48	16
1	49	17
2	50	18
3	51	19
4	52	20
5	53	21
6	54	22
7	55	23
8	56	24
9	57	25
10	58	26
11	59	27
12	60	28
13	61	29
14	62	30
15	63	31

To change Font#11 color to white, issue the following command: $VL04,59,FCFFFFFF*XX

Color index 96 through 255 are not used by Proteus and free to be used by the users.

6.4 RECTANGLE

6.4.1 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`	

6.4.2 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`	

6.5 STRING

6.5.1 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	

6.5.2 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	

6.5.3 DRAW

Draw and justify string within a rectangle area. Rectangle area can be filled with an alpha-blended color (Bcolor).

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

Examples:

13	X	Y	W	H	9	1	2	16	1	0	Hello	LR justify text in rect area X,Y,W,H (Figure 1)
13	X	Y	0	0	5	1	2	16	1	0	World	Center text @ X,Y (Figure 2)
13	X	Y	0	0	0	1	2	16	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	Select Font	*0..15*
Bcolor	Fill rectangle area defined by X,Y,W,H with this color index. Bcolor 0 = No fill	*0 and 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(optional)	If a string is provided, String #nn will be ignored.	-

Example

```
$VL14,300,100,400,500,9,1,2,16,1,0,Hello*XX  // Print 'Hello', L1, Font 2, LR justify in a Rect area(300,100,400,500)
$VL14,300,100,  0,  0,5,1,2,17,1,0,World*XX  // Print 'World', L1, Font 2, center text @(300,100)
$VL14,300,100,  0,  0,0,1,2,18,1,0,Proteus*XX// Print 'Proteus',     L1, Font 2, start printing text @(300,100)
```

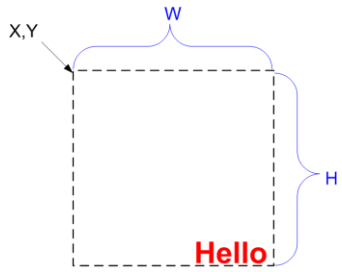


Figure 1

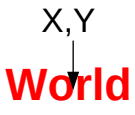


Figure 2

Figure 3

6.5.4 DRAW

Draw and justify string within a rectangle area. Rectangle area remains transparent.
This command will be deprecated in the future. Please use Command 13 instead.

Cmnd ID	Payload										
14	X	Y	W	H	Justify	Layer	Font	CLR	String ID	String (optional)	Description

Examples:

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

Payload Parm	Description	Range
X	Virtual rectangle area in which to justify the string. Rectangle area is transparent.	*0..1920*
Y		*0..1080*
W		*0..1920*
H		*0..1080*
Justify	Justification code	*Justify*
Layer	Destination Layer	Layers
Font	Select a Font	*0..15*
CLR	1 = Clear rectangle area before printing string, 0 = Add text to 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

```
$VL14,300,100,400,500,9,1,2,1,0,Hello*XX   // Print 'Hello',    L1, Font 2, LR justify in a Rect area(300,100,400,500)
$VL14,300,100,  0,  0,5,1,2,1,0,World*XX   // Print 'World',    L1, Font 2, center text @(300,100)
$VL14,300,100,  0,  0,0,1,2,1,0,Proteus*XX // Print 'Proteus',  L1, Font 2, start printing text @(300,100)
```

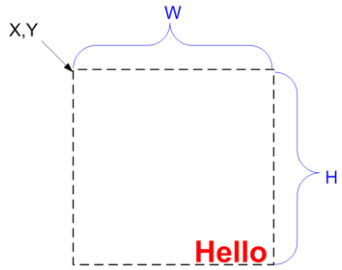


Figure 1

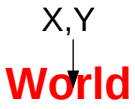


Figure 2

Figure 3

6.6 DRAW

6.6.1 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		

6.6.2 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		

6.6.3 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`		

6.6.4 SQUARE

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

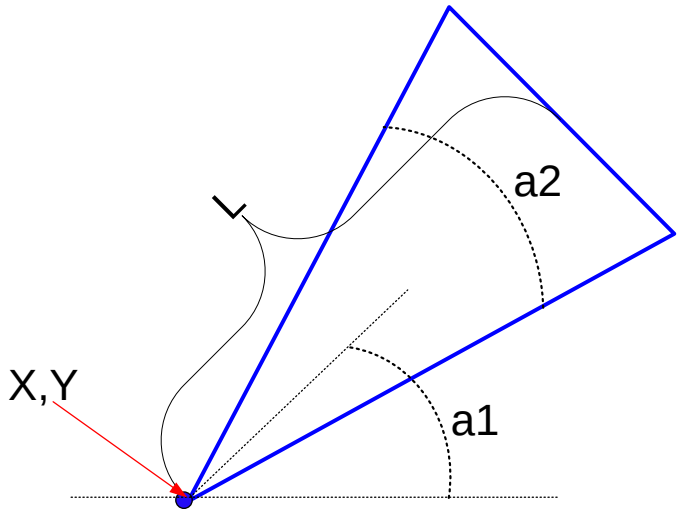
Payloads	Description	Range
X		*0..1920*
Y		*0..1080*
W		*0..1920*
H		*0..1080*
Color	Line color index	0..255
Layer	Layer	Layers
0	Unused	*0*
Size	Size of the line if square is not filled	*0..15*
Example		
`$VL55,100,100,1000,300,26,1,1,15*XX`		

6.6.5 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		



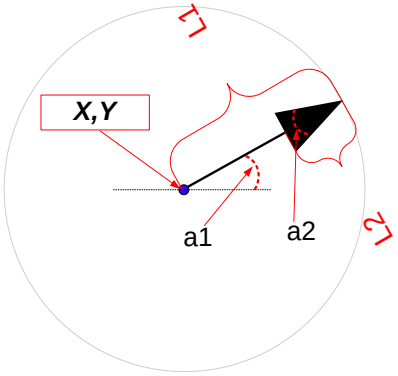
6.6.6 VECTOR

Draw a vector (line with an arrow).

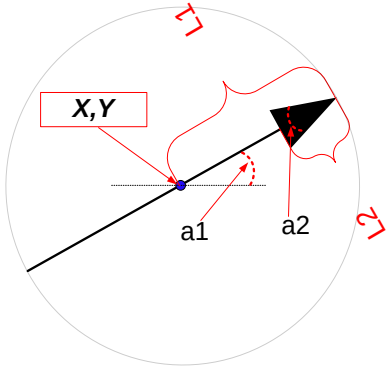
Cmnd ID	Payload									
53	X	Y	L1	a1	L2	a2	Color	Layer	Size	Type

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`	

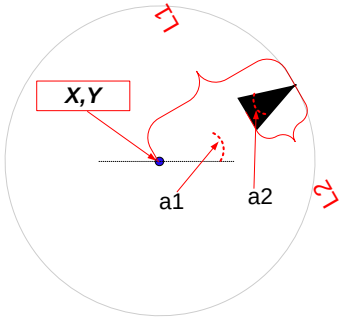
Type 0



Type 1



Type 2



6.6.7 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

```
$VL48,50,50,1,35*XX                    // X pixel/div =50, Y pixel/div = 50
$VL48,100,100,0,31*XX                  // X pixel/div =25, Y pixel/div =100
```

6.7 MACRO

6.7.1 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	Command stream

Payloads	Description	Range
Macro ID	Assigned ID	0..95
Command stream	Group of commands	ASCII

The **commandstream**[] array contains commands to be executed. Each command is composed of ID + Payload followed by '@' character.

Example1:

We would like to assign the following five commands to Macro 77:

Commands	*Description*	*Arguments*
Command 1	Clear Layer 0	**$VL01,0**
Command 2	Paint Rectangular	**$VL07,0100,0100,0200,0200,0,1247*XX**
Command 3	Display Image	**$VL25,10,0100,0100,0*XX**
Command 4	Draw String	**$VL13,00,04,0100,0100,1,Hello*XX**
Command 5	Paint Rectangle	**$VL07,0100,0100,0200,0200,1,794A*XX**

The command ID + payload would be constructed as follow:

$VL20,77,01,0@07,0100,0100,0200,0200,1,1247@**25,10,0100,0100,0@**13,00,04,0100,0100,1,Hello@**07,0100,0100,0200,0200,1,794A@***XX

To run Macro 77, issue Macro Execute command **$VL21,77***XX

Example2:

We would like to assign the same commands to Macro 77. However, when executing this macro, we would like to pass the X-coordinate of command 3 and the color of command 5 as arguments. To accomplish this, both parameters should be replaced with character '^' as shown below:

Commands	Description	Arguments
Command 1	Clear Layer 0	**$VL01,0**
Command 2	Paint Rectangle	$VL07,0100,0100,0200,0200,1,1247*XX
Command 3	Display Image	**$VL25,10,^^^^^0100,0*XX**
Command 4	Draw This String	**$VL13,00,04,0100,0100,1,Hello*XX**
Command 5	Paint Rectangle	**$VL07,0100,0100,0200,0200,1,^^^^*XX**

The command ID + payload would be constructed as follow:

$VL**20,77,01,0@**07,0100,0100,0200,0200,1,1247@**25,10,^^^^^0100,0@13,00,04,0100,0100,1,Hello@07,0100,0100,0200,0200,1,^^^^@***XX

To run Macro 77, issue Macro Execute command with associated arguments $VL**21,77,0200,794A***XX

6.7.2 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'	

6.7.3 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		

6.8 IMAGE

6.8.1 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	

6.8.2 DELETE

Delete a single image or a series of sequential images from FLASH.

Cmnd ID	Payload	
26	IDm	IDn

Payloads	Description	Range
IDm	First Image to be deleted	*00..95*
IDn	Last Image to be deleted	*00..95*
Example		
$VL26,10,10*XX	// Delete Image 10	
$VL26,10,12*XX	// Delete Images 10,11,12	

6.8.3 SAVE

This command is provided for users who wish to download images using their own application program. Proteus is supplied with a Windows App that facilitates storing images into FLASH.

Cmnd ID	Payload						
91	ID	x	y	Image Size	Image Name length	Image Name	Image data

Payloads	Description	Range	XMT as
ID	First Image to be deleted	*0..95*	*2 ASCII char*
X	Immediately after power up, image is load in the layer 3, virtual location X & Y. If X=0 and Y=0, image is autoloaded in the next available location.	*0..1920*	*4 ASCII char*
Y		*1080..2160*	*4 ASCII char*
Image Size	Size of the image in bytes	-	*8 ASCII char*
Image Name length	Number if characters in the image name	*0..31*	*2 ASCII char*
Image Name	Name of the image right justified.	-	*32 ASCII char*
Image data	Last Image to be deleted	-	*n ASCII HEX*

Example

```
$VL91,ii,XXXX,YYYY,SSSSSSSS,LL,..........................ABC,##############################################.....

ii       = Image ID
XXXX     = X location
YYYY     = Y Location
SSSSSSSS = Images size
LL       = Number characters in the image name i.e. 3 for ABC
..............ABC = Image Name
######...= Image data
```

6.9 GET STATUS

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

Reply will be three ASCII characters as shown below:

Reply ID	Payload			
35	Video Format	watchdog	Cold Boot	Version

Parm	Description		Range
Video Format	SD_UNKNOWN	10	10..46
	SD_PAL	11	
	SD_NTSC	12	
	INTERNAL	14	
	SDI_UNKNOWN	15	
	SDI_525i	16	
	SDI_625i	17	
	SDI_720p_50HZ	47	
	SDI_720p_60HZ	18	
	SDI_1080i_60H	19	
	SDI_1080i_50H	20	
	SDI_1080p_30H	21	
	SDI_1080p_25H	22	
	SDI_1080p_24H	23	
	SDI_1080p_48H	24	
	HDMI_UNKNOWN	25	
	HDMI_525i	26	
	HDMI_625i	27	
	HDMI_720p_60H	28	
	HDMI_1080i_60	29	
	HDMI_1080i_50	30	
	HDMI_1080p_30	31	
	HDMI_1080p_25	32	
	HDMI_1080p_24	33	
	HDMI_1080p_48	34	

	CP_UNKNOWN 35 CP_525i 36 CP_625i 37 CP_525p 38 CP_625p 39 CP_720p_60HZ 40 CP_720p_50HZ 41 CP_1080i_60HZ 42 CP_1080i_50HZ 43 CP_1080p_30HZ 44 CP_1080p_25HZ 45 CP_1080p_24HZ 46	
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	
Example		
<STX>35,45,0,1,V63.00,CS<ETX> // Video 1080p, watchdog is disabled, Proteus lost power since last status request <STX>35,42,1,0,V63.01,CS<ETX> // Video 1080i, watchdog is enabled, Proteus has not lost power since last status request		

6.10 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 reister are updated automatically when the content of the register changes.

Register ID	Contents	Description
3	*Video Mode SD*	
4	*Video Mode HD*	
5	*UTC Offset*	*-HH:MM*
6	*Proteus RTC Time*	*Time & Date from Proteus built-in clock*
7	*Proteus RTC Date*	
8	*IRIG-B RTC Time*	*External IRIB-B source*
9	*IRIG-B RTC Date*	
10	*Video Ancillary Time Code*	
11	*Video Ancillary Date Code*	
12	*Count Down Timer*	*HH:MM:SS, MM:SS, SS, HH:MM:SS:FF*
13	*Count Up Timer*	*HH:MM:SS, MM:SS, SS, HH:MM:SS:FF*
14	*Analog Input 1 : Raw$_1$*	
15	*Analog Input 2 : Raw$_2$*	
16	*Analog Input 3 : Raw$_3$*	
17	*Analog Input 4 : Raw$_4$*	
18	*Analog Input 5 : Raw$_5$*	
19	*Analog Input 6 : Raw$_6$*	
20	*Analog Input 7 : Raw$_7$*	
21	*Analog Input 8 : Raw$_8$*	
22	*Analog Input 1 : Map$_1$*	
23	*Analog Input 2 : Map$_2$*	
24	*Analog Input 3 : Map$_3$*	
25	*Analog Input 4 : Map$_4$*	

26	*Analog Input 5 : Map$_5$*	
27	*Analog Input 6 : Map$_6$*	
28	*Analog Input 7: Map$_7$*	
29	*Analog Input 8: Map$_8$*	
30	*Analog Input 1: Raw Differential$_1$*	
31	*Analog Input 2: Raw Differential$_2$*	
32	*Analog Input 3: Raw Differential$_3$*	
33	*Analog Input 4: Raw Differential$_4$*	
34	*Analog Input 1: Map Differential$_1$*	
35	*Analog Input 2: Map Differential$_2$*	
36	*Analog Input 3: Map Differential$_3$*	
37	*Analog Input 4: Map Differential$_4$*	
38	*Counter 1: Raw$_1$*	Counters (Quadrature or Simple)
39	*Counter 2: Raw$_2$*	
40		
41		
42	*Counter 1: Map$_1$*	
43	*Counter 2: Map$_2$*	
44		
45		
46	*INPUT1..8*	*State of digital inputs*
47		
48		
49		
50		
51	*Token-HeaderA*	
52	*TokenA1*	
53	*TokenA2*	
54	*TokenA3*	
55	*TokenA4*	
56	*TokenA5*	

57	*TokenA6*	*Parameters read from a CSV sentence A*
58	*TokenA7*	
59	*TokenA8*	
60	*TokenA9*	
61	*TokenA10*	
62	*TokenA11*	
63	*TokenA12*	
64	*Token-HeaderB*	*Parameters read from a CSV sentence B*
65	*TokenB1*	
66	*TokenB2*	
67	*TokenB3*	
68	*TokenB4*	
69	*TokenB5*	
70	*TokenB6*	
71	*TokenB7*	
72	*TokenB8*	
73	*Token-HeaderC*	*Parameters read from a CSV sentence C*
74	*TokenC1*	
75	*TokenC2*	
76	*TokenC3*	
77	*TokenC4*	
78	*TokenC5*	
79	*TokenC6*	
80	*TokenC7*	
81	*TokenC8*	
82	*Reticle1 X*	
83	*Reticle1 Y*	
84	*Reticle2 X*	
85	*Reticle2 Y*	
86	*Reticle3 X*	
87	*Reticle3 Y*	
88	*Reticle4 X*	

89	*Reticle4 Y*	
90	*Marker X1*	*Data associate with XY measurements*
91	*Marker X2*	
92	*Marker Y1*	
93	*Marker Y2*	
94	*Marker Δx*	
95	*Marker Δy*	
96	*Marker Mapped_Δx*	
97	*Marker Mapped_Δy*	
98	*BMP: Pressure*	
99	*BMP: Temerature*	
100	*BMP: Altitude*	
101	***GPS COM1,4**: Altitude*	*Parameters read from GPS Modem#1 connected to port COM1 or COM4*
102	***GPS COM1,4**: Coarse(heading) Over Ground*	
103	***GPS COM1,4**: Speed Over Ground*	
104	***GPS COM1,4**: Time*	
105	***GPS COM1,4**: Date*	
106	***GPS COM1,4**: Latitude (±dd.ddddd)*	
107	***GPS COM1,4**: Longitude(±ddd.ddddd)*	
108	***GPS COM1,4**: Latitude (dd° mm' ss.s")*	
109	***GPS COM1,4**: Longitude (dd° mm' ss.s")*	
110	***GPS COM2,3**: Altitude*	*Parameters read from GPS Modem#2 connected to port COM2 or COM3*
111	***GPS COM2,3**: Coarse(heading) Over Ground*	
112	***GPS COM2,3**: Speed Over Ground*	
113	***GPS COM2,3**: Time*	
114	***GPS COM2,3**: Date*	
115	***GPS COM2,3**: Latitude (±dd.ddddd)*	
116	***GPS COM2,3**: Longitude(±ddd.ddddd)*	
117	***GPS COM2,3**: Latitude (dd° mm' ss.s")*	
118	***GPS COM2,3**: Longitude (dd° mm' ss.s")*	
119	***INS**: Heading*	
120	***INS**: Pitch*	

121	***INS**: Roll*	
122	***INS**: Altitude*	
123	***INS**: Latitude (±dd.ddddd)*	
124	***INS**: Longitude (±ddd.ddddd)*	
125	***INS**: Time*	
126	***INS**: Date*	
127	***INS**: Latitude (dd° mm' ss.s")*	
128	***INS**: Longitude (dd° mm' ss.s")*	
129	**$SDDBT** *: Depth (m)*	
130	**$SDDPT:** *Water Depth relative to transducer(m)*	
131	**$SDDPT:** *Offset from transducer (m)*	
132	**$SDDPT:** *Maximum range scale in use*	
133	**$WIMTW:** *Water Temperature in C*	

6.10.1 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	

6.10.2 SET REGISTER

Set a register (range 3..127) to a desire value. The Widget that use this 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	

6.11 REAL TIME CLOCK

6.11.1 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	

6.11.2 SET DATE

Cmnd ID	Payload
45	Date

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

6.11.3 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	

6.11.4 UP/DOWN TIMER

This command allows user to clear, preset, start or stop the real-time timer.

Cmnd ID		Payload			
62	Timer#	Up/Down	Display Format	Control	Preset

Parm	Description	Range
Timer #	0 = Select timer #1 1 = Select timer #2	0..1
Up/Down	0 = Count down 1 = Count up	0..1
Display Format	0 = HH:MM:SS 1 = MM:SS (could become MMM:SS) 2 = SS (Could become SSS)	0..2
Control	0 = Pause timer 1 = Resume timer 2 = Toggle (Start/Stop) timer (ON ↔OFF)	0..3
Flash	0 = Flash when 0, 1 = Do not flash	0..1
Preset	If value HH:MM:SS is present, timer# = HH:MM:SS	*HH:MM:SS*

Example

```
$VL62,0,1,0,0,0*XX              // timer#1, Count up,   HH:MM:SS,  Pause
$VL62,1,1,1,1,0*XX              // timer#2, Count up,     MMM:SS,  Resume
$VL62,1,1,0,2,0*XX              // timer#2, Count up,   HH:MM:SS,  Pause ↔ Resume
$VL62,0,0,2,0,0,01:45:00*XX     // timer#1, Count down, HH:MM:SS,  Pause              ,Preset to 01:45:00
$VL62,0,1,2,0,0,00:00:00*XX     // timer#1, Count up,   HH:MM:SS,  Pause              ,Preset to 00:00:00
```

Regardless of the "Display Format", the preset value should always be in the HH:MM:SS format.

6.12 EMULATE PS2 KEYBOARD

This command allows the user to emulate PS2 keyboard through RS232 commands

Cmnd ID	Payload
56	Scan code

Parm	Description	Range
PS2 Scan code	-	0.FF
Example		
$VL56,05*XX	// Emulate F1	
$VL56,06*XX	// Emulate F2	
$VL56,04*XX	// Emulate F3	
$VL56,37*XX	// Emulate 7	
$VL56,78*XX	// Emulate F11	
$VL56,1B*XX	// Emulate ESC	