HAIDAR TECHNOLOGY, LLC.

The Next Generation Of Intelligent Embedded GUI Systems

www.HaidarTechnology.com
Phone: (614) 389-3022
Fax : (614) 923-8668
Sales@haidartechnology.com

SegeNT

Serial-Enabled Graphic Engine

Firmware Version 1.1na

Software Reference Manual

REV 1.2

Revision 1.2
Issue Date: 3/24/2014

Important Notice:
Haidar Technology products are not designed, intended, authorized, or warranted to be suitable for use in life-support applications, devices, or systems, or in other critical applications.
Haidar Technology and the buyer agree that Haidar Technology will not be liable for incidental or consequential damages arising from the use of Haidar Technology products. It is the user's responsibility to protect life and property against incidental failure. Haidar Technology reserves the right to make changes and improvements to its products without providing notice

Contents

1. Overview: 7
2. Features: 7
3. Typical Applications: 7
4. SegeNT Block Diagram: 8
5. Main and Graphic Controller: 8
7. LCD Timing Parameters: 9
8. Graphic Layers: 9
10. Serial Interface: 10
11. Communication Protocol: 11
12. Touch Screen Interface: 11
13. Bitmaps: 12
14. Fonts: 12
15. Audio: 14
16. Flash Memory Map: 15
17. Project Hex File: 16
18. Flash Programming using the serial interface: 17
19. Flash Programming from the micro SD card: 17
20. TERMINAL COMMANDS: 18

18.2. Get_FirmWareVersion 20
18.3. Get_SerialNumber 20
18.4. Get_UserVersion 20
18.5. Get_UserIDNumber0 20
18.6. Get_UserIDNumber1 21
18.7. Get_LCDSize 21

18.8. Get_TouchScreenType 21
18.9. Get_FlashMemorySize 21
18.10. Get_UserFlashMemorySize 22
18.11. Get_SystemStatus 22
18.12. Cal_TouchScreen 22
18.13. Set_Brightness 23
18.14. Set_Backlight 23
18.15. Set_Beep 23
18.16. Set_LCD 23
18.17. Write_UserFlashMemory 24
18.18. Read_UserFlashMemory 24
18.19. Erase_UserFlashMemory 24
18.20. Set_DigitalOutput 24
18.21. Get_DigitalInput 24
18.22. Get_AnalogInput 25
18.23. Set_PWM 25
18.24. Get_TouchData 25
18.25. Get_ProductCode 25
18.26. Play_Audio 25
18.27. Set_EntVolume 26
18.28. Set_ExtVolume 26
18.29. Mute 26
18.30. Clear_Screen 26
18.31. Set_BackColor 26
18.32. Set_DrawColor 27
18.33. Set_DrawWidth 27
18.34. Set_DrawType 27
18.35. Draw_Line 27
18.36. Draw_Rectangle 28
18.37. Draw_Bevel 28
18.38. Draw_Circle 28
18.39. Draw_Pixel 28
18.40. Draw_FilledBevel 29
18.41. Draw_FilledCircle 29
18.42. Draw_FilledRectangle 29

18.43. Copy_Area 30
18.44. Paste_Area 30
18.45. Get_Pixel 30
18.46. Set_ActiveLayer 30
18.47. Get_ActiveLayer 31
18.48. Get_CopyBuffer 31
18.49. Draw_Bitmap 31
18.50. Show_Bitmap 32
18.51. Get_BitmapDimension 32
18.52. Get_BitmapBackStyle 32
18.53. Set_Font 33
18.54. Write_Text 33
18.55. Get_TextWidth 33
18.56. Get_TextHeight 33
18.57. Get_TextLength 34
18.58. Get_TextLineCount 34
18.59. Get_FontHeight 34
18.60. Get_FontWidth 34
18.61. Get_Font 35
18.62. Program_Headers 35
18.63. Program_GUI 35
18.64. Program_Bitmaps 35
18.65. Program_Fonts 36
18.66. Program_Audio 36
18.67. Erase_FlashMemory 36
18.68. Erase_Headers 36
18.69. Erase_GUI 37
18.70. Erase_Fonts 37
18.71. Erase_Bitmaps 37
18.72. Erase_Audio 37
19. Graphical User Interface (GUI) 39
19.1 Object Code: 39
19.2 Object ID 39
19.3 Input Interface (Touch Event): 40
19.5 User Interface Message (uiMessage): 41

19.6 Object Properties: 41
19.7 Object Standard Colors: 42
19.8 GUI Buffer: 42
19.9 Object Draw Priority: 43
19.10 Object UI Priority: 43
20. Graphical User Interface Objects: 44
20.2. TextBox: 46
20.3. Button: 48
20.4. Needle: 51
20.5. NumberBox: 53
20.6. Slider: 56
20.7. Chart: 58
20.8. BarGraph: 60
20.9. PictureBox: 62
20.10. Image: 64
20.11. Shape: 67
21. GUI COMMANDS 69
21.1. Form_Load 70
21.2. Form_Draw 70
21.3. Form_View 70
21.4. Form_Move 70
21.5. Get_ActiveScrID 71
21.6. Get_ActiveWin1ID 71
21.7. Get_ActiveWin2ID 71
21.8. Enable_UI 71
21.9. Form_Enable 71
21.10. Get_uiMessage 72
21.11. Get_uiStatus 72
21.12. Form_Status 72
21.13. Get_ObjProperty 73
21.14. Set_ObjProperty 73
21.15. From_Redraw 74
21.16. Redraw_Object 74
21.17. Chart_AddPoint 75
21.18. Chart_Clear 75

21.19. Chart_VCL 75
21.20. Chart_HCL 75
21.21. TextBox_Clear 76
21.22. TextBox_State 76
21.23. TextBox_Text 76
21.24. TextBox_Number 76
21.25. BarGraph_Value 77
21.26. Button_State 77
21.27. NumberBox_Value 77
21.28. Needle_Value 78
21.29. Slider_Value 78
21.30. Image_Value 78
21.31. Image_Play 78
21.32. Image_Stop 78
21.33. PictureBox_Clear 79
21.34. PictureBox_Bitmap 79
21.35. PictureBox_Line 79
21.36. PictureBox_Rectangle 79
21.37. PictureBox_Circle 80
21.38. PictureBox_SetPixel 80
21.39. PictureBox_GetPixel 80
21.40. PictureBox_CopyArea 80
21.41. PictureBox_PasteArea 81
21.42. PictureBox_Print 81
21.43. PictureBox_Picture 81

ASCII Character Codes: 83

Basic Colors: 88

Manual Change History: 94

Hardware Limited Warranty 94

Returns and Repair Policy 94

1. Overview:

SegeNT is the second generation of Haidar's successful family of Graphical User Interface (GUI) controller boards for color TFT displays. SegeNT is hardware compatible with SegeMax and offer more features and capabilities.

GUI controllers provide visual feedback via an LCD and user input functions via touch screen and act as the intelligent interface between human and machine. SegeNT uses a serial (UART) interface for host communication and offer a rich command set that provides access to high-level graphics primitive drawing and touch screen event handling capabilities, offloading such complex tasks from the host processor or embedded controller.

SegeNT uses the same techniques used in objective oriented programming to create the embedded GUI without any additional code for the LCD or the touch panel. A powerful set of objects and a window based software "uiLAB" are used to make this task extremely easy and fast.

uiLAB is a true WYS|WYG visual GUI builder for color displays. It allows you to design the GUI application visually from your PC screen using simple Drag-and-Drop tools with absolutely no coding for the GUI design.

2. Features:

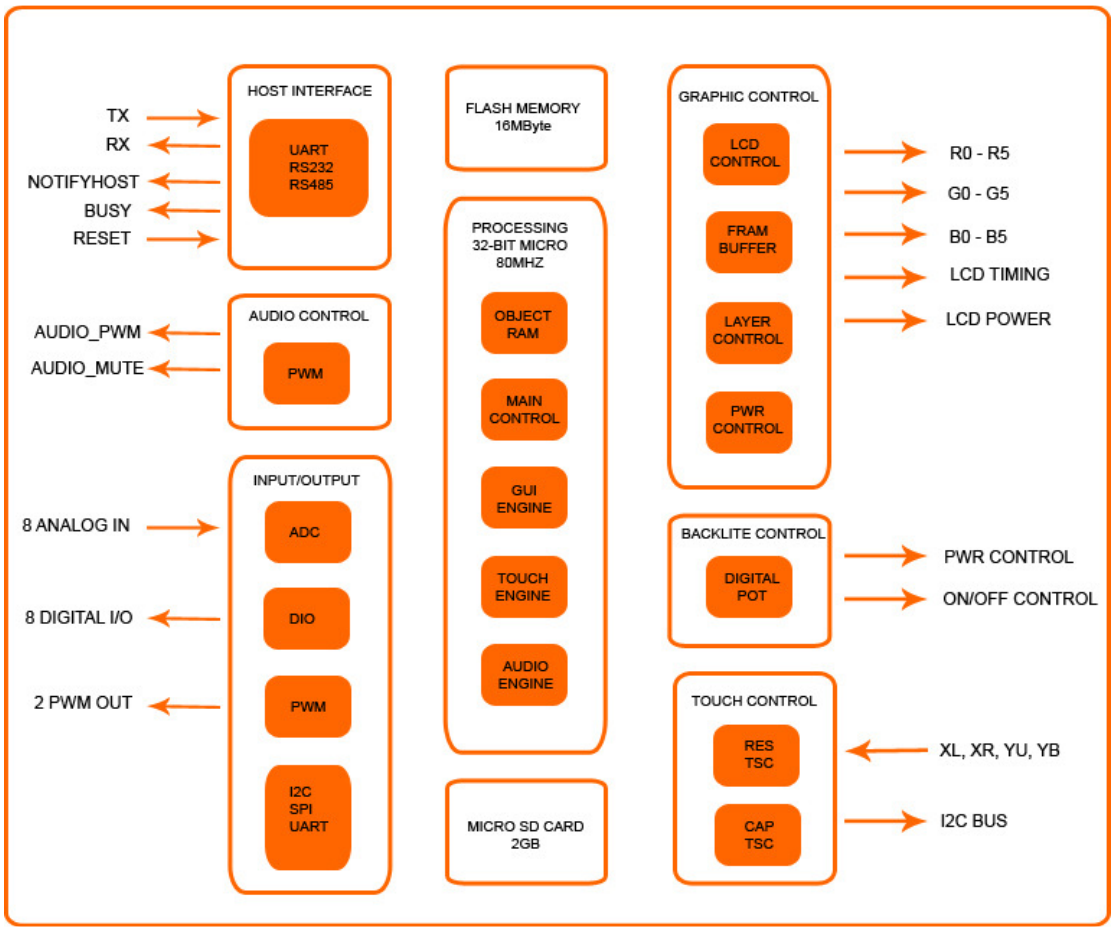
- Powerful set of GUI Objects, including: Form, Button, BarGraph, Chart, NumberBox, TextBox, PictureBox, Slider, Needle and Shape.
- Advanced Object Oriented Architecture enables low cost MCU as system host using UART (RS232) interface.
- Supports TFT color LCD displays of various sizes and up to WVGA (800X480).
- True color or 16 bit per pixel (bpp) color depth.
- Three independent graphic layers (Screen, Window1 and Window2) support.
- Alpha blending and transparency support.
- Window's Bitmap support with rotation and mirror.
- Anti-Aliased (2bpp) and normal (1bpp) fonts support.
- SD card FA16 file system support. GUI project can be programmed directly from the SD card.
- 64KB Flash memory for user application.
- Mono Audio channel with PWM output.
- Built-in Sound Generator.
- User defined Audio Play Back files.
- Resistive and Capacitive touch screen support.
- Back light brightness control.
- Digital I/O, Analog inputs and PWM support.
- 16Mbyte flash memory for GUI, fonts and bitmaps storage.
- Simple UART serial interface.
- No missing touch events!
- Fast graphic rendering.

3. Typical Applications:

- Instrumentation
- Industrial Control
- Medical Devices
- Home Appliances
- Vending Machines
- …. And many more

4. SegeNT Block Diagram:

Below is the block diagram of the SegeNT board and the typical interface to the host controller. The basic interface only requires UART Rx and Tx to interface your controller to the SegeNT board.



Upon arrival, the command and its data stored in the command buffer, and then processed by the command interpreter to be executed. Once the controller executes the command, it will send an ACK/NAK code depends on if the command executed successively or not.

The host controller needs to wait until it receives the ACK/NAK code before sending another command to the controller (simple ACK/NAK interface). This will prevent the host from jamming the interface and filling the command buffer and also keep the host and LCD controller in synch.

Please see SegeNT Hardware manual for more information on how to interface your host controller to SegeNT.

5. Main and Graphic Controller:

SegeNT is based on PIC32MX795F512L microcontroller from Microchip. It is a powerful 32-bit processor with 512KB flash memory and 128KB of RAM. The fast processing speed (80MHZ) and the huge RAM, making it an ideal choice to handle all the function of SegeNT.

The graphic LCD controller on board is S1D13748 from Epson. It is a powerful controller with 1024KB of frame buffer and three layers of graphic. It is capable of driving TFT displays up to 800X480 at 16 bbp color depth or 65536 colors. S1D13748 generates all the necessary signals to drive the LCD, which includes PCLK, VSYNCH, HSYNCH, ENABLE, R0-R5, G0-G6 and B0-B5. Alpha blending and transparency is also supported between the three graphic layers.

6. Colors:

The pixel color is stored in the frame buffer in RGB565 format, which is a 16-bit value with 5 bits for Red, 6 bits for Green and 5 bits for Blue.
The host controller must send the color data in the RGB565 format for the color to be displayed correctly on the LCD. The following macro can be used by the host to convert the standard RGB888 format to RGB565 format:

```
#define  RGB565CONVERT(red, green, blue)\
(WORD)((( red >> 3 ) << 11 ) | (( green >> 2 ) << 5  ) | ( blue  >> 3 ))
```

7. LCD Timing Parameters:

The LCD timing parameters, which are provided by the manufacturer in the data sheet of the display, are very important to drive the display. The LCD timing parameters are saved in the flash memory under the configuration header and called back at power on or reset. The LCD timing values are calculated at power on or reset and then sent to the graphic controller. Please note that the LCD will not initialize if the parameters are not saved correctly and this may damage the LCD. You can set the LCD timing parameters using uiLAB (Config->Hardware->LCD).
The table below shows the most important LCD timing parameters used by SegeNT

Parameter	Description
Horizontal Back Porch (HBP)	Number of PIXCLK cycles between the HSYNC signal and first valid pixel data
Horizontal Front Porch (HFP)	Number of PIXCLK cycles between the last valid pixel data in the line and the next HSYNC pulse
Vertical Back Porch (VBP)	Number of lines (HSYNC pulses) between the asserted VSYNC signal and the next valid line
Vertical Front Porch (VFP)	Number of lines (HSYNC pulses) between the last valid line of the frame and the next VSYNC pulse
VSYNC pulse width (VPW)	Number of lines (HSYNC pulses) between the last valid line of the frame and the next VSYNC pulse
HSYNC pulse width (HPW)	Number of PIXCLK pulses when the HSYNC signal is active
VSYNC Start Position (VPSP)	It is the number of lines before VSYCN starts
HSYNC Start Position (HPSP)	It is the number of Pixels (PCLK) before HSYNC starts
Active frame width (LCDResX)	Horizontal resolution, which is the number of pixels in a line.
Active frame height (LCDResY)	Vertical resolution of the LCD. For a WVGA display of resolution, 800H × 480V, the value of the frame height is 480 lines.
VSYNC polarity	It is the value held by the VSYNC signal to indicate the beginning of a new frame. The VSYNC polarity can be active low or active high.
HSYNC polarity	It is the value held by the HSYNC signal to indicate the beginning of a new line. The HSYNC polarity can be active low or active high.
PCLK polarity	This determine if the RGB data will be latched at the rising or falling edge of PCLK
DE polarity	This determine if ENAB or DE is active at high or low state

8. Graphic Layers:

SegeNT has three layers of graphic:

- Screen (main) Layer: This layer has the same resolution (size) as the display and only displayed at (0, 0). It is not movable or sizable but it can be viewed or hidden. If it is hidden then the back ground color (black) will be displayed.
- Window1 (PIP1) Layer: The maximum size of this layer depends on the display resolution. The width or height must be greater than 2, divisible by 2 and must be smaller than 512. This layer is movable and it can be viewed or hidden without affecting the background image.
- Window2 (PIP2) Layer: The maximum size of this layer depends on the display resolution. The width or height must be greater than 2, divisible by 2 and must be smaller than 512. This layer is movable and it can be viewed or hidden without affecting the background image.

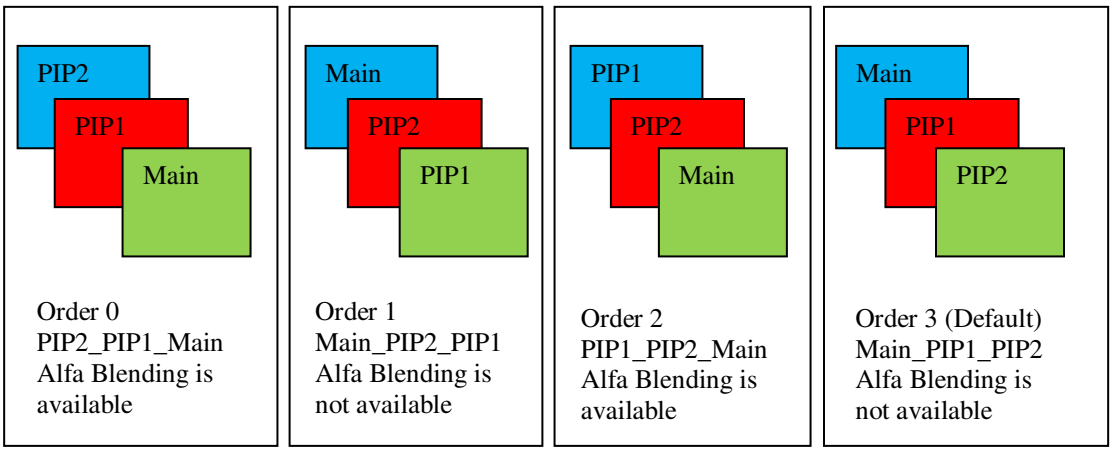
The layer size = (2*width*Height)/1024 in KB.
The table below shows the maximum size for each layer:

Display Resolution	Max Screen Size	Max Window1 Size	Max Window2 Size
QVGA (320X240)	150 KB	150 KB	150 KB
WQVGA (480X272)	255 KB	255 KB	255 KB
VGA (640X480)	600 KB	350 KB	74KB
WVGA (800X480)	750 KB	200 KB	74 KB

Three layers (screen, window1 and window2) can be displayed at the same time without the need to modify the background image. The user interface (touch events) for each layer can be enabled or disabled at design and run time.

9. Alpha Blending and Transparency:

The Layer Display Order determines how the layers will be shown on the display area. The figure below shows the four possibilities:



Alfa blending is only available if the layer display order is 0 or 2. Transparency is available regardless of the layer display order. The default order is Main_PIP1_PIP2 which ensure showing all layers regardless of the Alfa blending settings. Please note that Order 0 or 2 will hide PIP2 and PIP1 behind the Main layer and you will not able to see them if the Alfa blending is not enable or set correctly.
The Alfa blending can be performed between the three layers or only between PIP1 and PIP2. Four independently enabled key colors are available. Each key color can support a blending ratio from 0% to 100% and in 12.5% steps.
The main layer must have at least one area with the enabled key color and the PIP layer must overlap with this area for the Alfa blending to work with good results.
Transparency is only available for PIP1 and PIP2. Each PIP layer has one transparency color, which will not be displayed if the transparency is enabled for the layer.

10. Serial Interface:

UART (CMOS level at 3.3V)
Baud Rate: User defined 19200, 38400, 57900 and 115200 bps (Default)
Number of data bits: 8
Number of stop bits: 1
Parity: OFF
Handshacking: None
Communication address or Device ID: User defined from 1 to 254 (Default = 16)
Transmitter Enable Signal (Only for RS485)

Warning: RX and TX use a CMOS level of 3.3V. Connecting them to standard (PC) RS232 with +/- 12V or other will damage the controller and void your warranty.

11. Communication Protocol:

The interface between SegeNT and the host controller is a simple ACK/NAK protocol where the host is always the master and SegeNT is the slave. After a command has been received and processed, SegeNT will send a response packet to the host. The first byte of this packet is the response code, which tells the host if the command was executed successively (ACK) or not (NAK). The host must wait to receive the response packet before sending another command. The command will be ignored if

1- The device ID is incorrect
2- Check sum error

Only the response code will be sent if the response NOB is zero.

Command Format

Device ID
NOB H
NOB L
Command Code
Data Field up to 256 bytes
Check Sum (CS)

Response Format

Response Code
NOB
Data Field up to 256 bytes
Check Sum (CS)

Where:
H: is the MS Byte
L: Is the LS Byte
NOB: is the number of bytes to follow excluding the check sum
Check Sum (CS): is the LSB of the summation of all the bytes in the packet

Response Code:

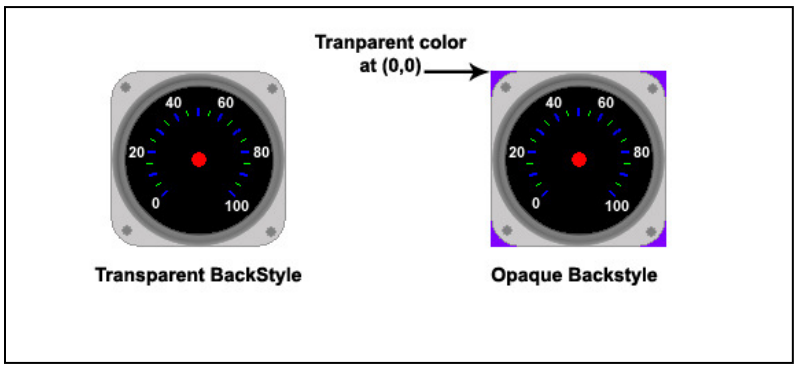
Response Code	Description
0x06 (6)	ACK
0x15 (21)	NAK

12. Touch Screen Interface:

SegeNT has a dedicated and powerful resistive (4-wire or 5-wire) touch screen controller (AR1021) from Microchip. It has sophisticated touch screen algorithms to process all touch data, saving the main controller form the processing overhead. 4-point touch screen calibration algorithm is used to calibrate the touch screen and provide corrected data to the main controller. The calibration points are saved to the AR1021 EEPROM. UiLab has a utility to calibrate and test the touch screen.
Touch screen calibration can be done by sending terminal command “CAL_TOUCHSCREEN” or by setting the dip-switch to mode 3. Calibration screen will be displayed and you just need to touch the red circles at each corner of the display.
SegeNT can also interface to capacitive touch panel with FT5x06 controller. Capacitive touch screen does not calibration and unlike resistive touch screen, one cannot use a capacitive touch screen through most types of electrically insulating material, such as gloves.
Touch sensitive area of the screen is defined as Touch Zone. Most objects can have a touch zone that when is pressed or released, a touch event will be generated to notify the host and/or to execute predefined macros.

13. Bitmaps:

The look and feel of the GUI is created by using standard window bitmaps (.bmp) which will be used for the screen background, buttons and controls. Bitmaps can be designed using any graphic design software like Adobe Photoshop. 14MB of the flash memory is assigned to save up to 512 bitmaps. The width and height of the bitmap **must be even number** and less than the display width or height. UiLAB, at compiling time, assign a unique ID number for each bitmap used in the project. The range of the ID number is form 0 to 511. The bitmap back style can be set to Opaque or Transparent. The transparent color is the color at (0, 0) and this color will not be displayed if the bitmap back style is set to transparent. Rotation (0, 90, 180 and 270) and mirror can be performed on a bitmap with back style set to opaque only.



14. Fonts:

SegeNT can have up to 8 user defined proportional fonts. uiLAB has a utility to convert a system font to predefined bitmaps, which can be saved to the flash memory. Only ANSI mode is supported in the current firmware. In ANSI mode, each font contain up to 244 characters (character codes from 32 to 255). 8 bit character codes (ASCII) are used to write text on the display.

Each font character is stored as contiguous set of bytes. Each pixel is represented by 1bit for normal fonts and by 2bits for anti-aliased fonts.

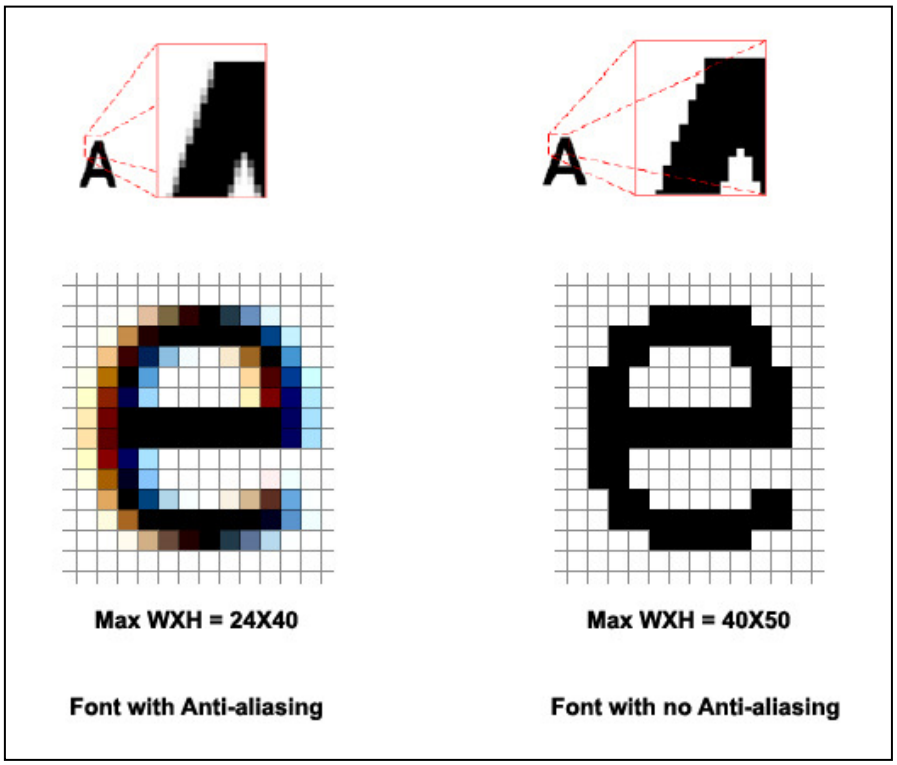
Each font character is defined as bitmap or matrix. The maximum bitmap width and height is 24X40 pixels for anti-aliased (2bpp color depth) fonts and 40X50 pixels for normal (1bpp color depth) fonts. The font character width or height should not exceed the maximum matrix width or height. The font character is placed according to certain fixed positions, which are determined during the conversion by the Font Converter utility.

In proportional writing, the cell width is equal to the character width and the cell height is equal to the font height. One pixel space is used between the characters. The font height and each character width are saved in the flash memory for each installed font. The firmware uses those parameters to determine the next character cell starting (x,y) position. Each font has a unique code that must be used by the host to set the current active font. After power up or reset, the current active font is Font0. The table below lists the font codes:

Font	Font ID	# Of Characters
Font0	0	244 (32 to 255)
Font1	1	244 (32 to 255)
Font2	2	244 (32 to 255)
Font3	3	244 (32 to 255)
Font4	4	244 (32 to 255)
Font5	5	244 (32 to 255)
Font6	6	244 (32 to 255)
Font7	7	244 (32 to 255)

Anti-aliasing is a technique used to make the edges of text appears smooth. This is useful, especially with characters such as ‘A’, ‘O’, etc., which have slanted or curved lines. Since the pixels of the display are arranged in rectangular fashion, slanted edges cannot be represented smoothly. To make them appear

smooth, a pixel adjacent to the slanted pixels is painted with an average of the foreground and background colors.
Since the average of foreground and background colors needs to be calculated at run-time, the rendering of anti-aliased fonts may take more time than rendering normal fonts. To optimize the rendering speed, anti-aliased fonts are only work over constant background color where character pixel color is calculated once while rendering each character.



15. Audio:

SegeNT provides mono audio output through PWM output pin "Audio_PWM". It outputs the built-in sounds and the user defined audio files.

The Audio Engine generates the sound effects from two sources:

- **Built-In sounds**: Up to 24 sound tables are stored at the microcontroller ROM and can be used to generate a click sound when an object is touched by the user. The ID range for the built-in sounds are from 16 to 39 as shown in the table below:

ID	Sound
16	DTMF0
17	DTMF1
18	DTMF2
19	DTMF3
20	DTMF4
21	DTMF5
22	DTMF6
23	DTMF7
24	DTMF8
25	DTMF9
26	DTMF*
27	DTMF#
28	Click-16A
29	Click-17A
30	Click-1A
31	Click-1B
32	Click-20A
33	Click-4C
34	Click-D
35	Click-O
36	Click-R
37	Click-T
38	Click-U
39	Switch

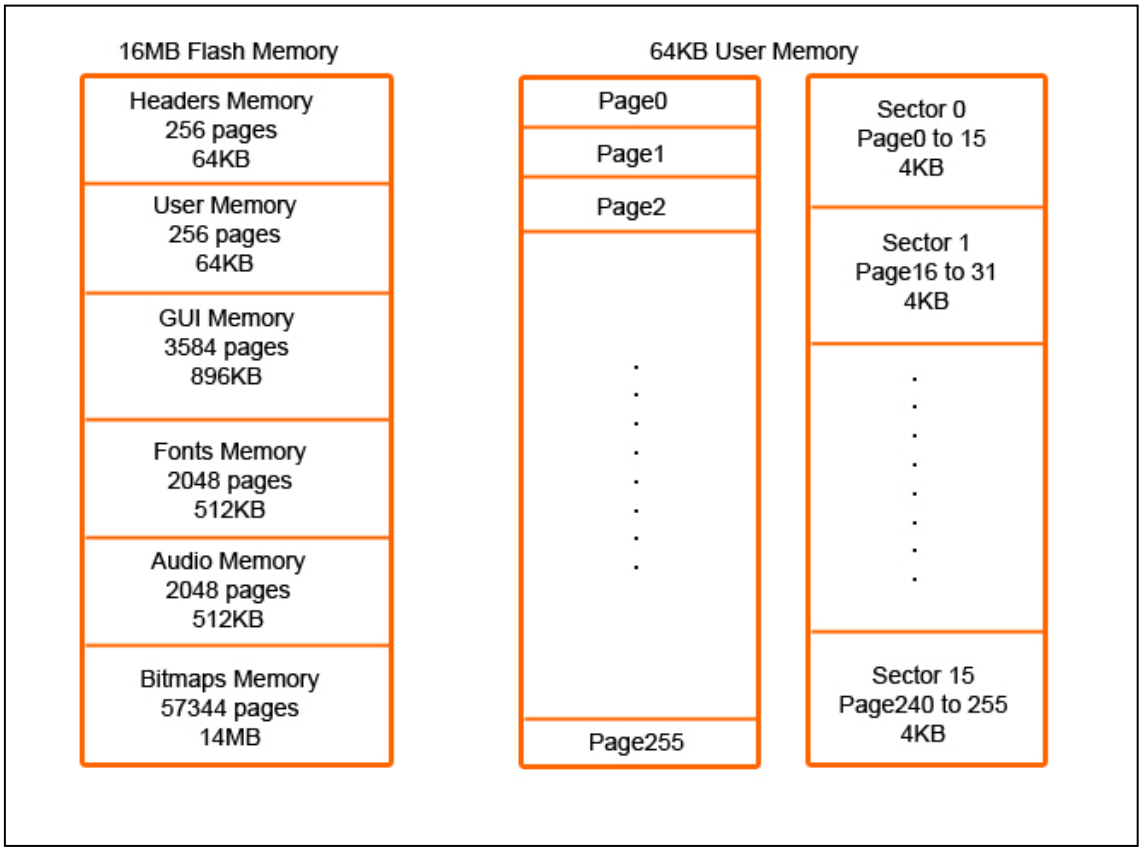
- **User-Defined Audio**: Up to 16 audio files can be edited and saved to the flash memory. uiLAB has a utility "Audio Converter" to convert standard Wave (.wav) audio files to uiLAB audio files (.uha) in which can be stored to the flash memory. The maximum size of the audio files is 512KB and the ID range is from 0 to 15.

The wave file (.wav) must be mono, 16-bit PCM and sample rate of 44.1kb.sec. Such files can be created or converted using commonly available audio editing programs like "Audacity". The audio converter utility converts the audio file to raw 8-bit signed that can be stored at the flash memory.

Two independent volume registers are used to adjust the audio volume. One for the built-in sounds and the second for the user defined audio. Both volumes can be set form 0 to 100. The default value for both is 50. A hardware mute signal is also provided to shut down the audio amplifier.

16. Flash Memory Map:

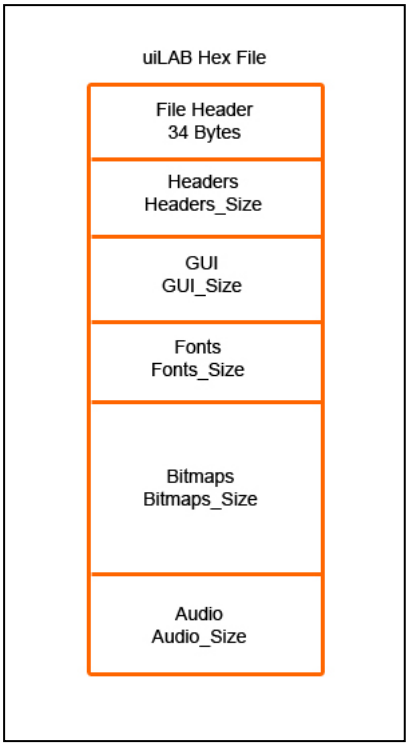
SegeNT has a 16MB flash memory to save the GUI project (objects) and its resources (fonts and bitmaps). It is organized into 65536 programmable pages of 256-bytes each. Up to 256 bytes can be programmed at a time. Pages can be erased in groups of 16 (4KB sector erase). Below is SegeNT flash memory map.



- **Headers Memory:** This section of the flash memory is used to save the project headers. Only the first 35pages are used and the remaining 221 pages are reserved for future use. The first header (256 bytes) is the Configuration header which has all the data necessary to configure SegeNT. The remaining 34 headers are for the project fonts, bitmaps and audio. At power up, the headers section will be saved to the internal RAM for fast access of the headers data.
- **User Memory:** This section of the flash memory is used by the user to save auxiliary data. Terminal commands Write_UserFlashMemory, Read_UserFlashMemory and Erase_UserFlashMemory can be used by the host to access this section. The host can read or write one full page (256 bytes) at a time. Before writing a page, the host needs to erase the sector which contains the page. The erase command can only erase on sector (16 pages) at a time.
- **GUI Memory:** This section of the flash memory is used to save the GUI project objects. Only the first 2112 pages are used which can save up to 64 forms (screen/window) and each form can have up to 64 objects.
- **Fonts Memory:** 2048 pages are reserved to save up to 8 fonts. Each font can have up to 244 characters.
- **Audio Memory:** 2048 pages are reserved to save up to 16 Audio files.
- **Bitmaps Memory:** 57344 pages are reserved to save up to 512 bitmaps at 16-bit color depth.

17. Project Hex File:

The Hex file is a binary file generated by uiLAB after a successful build or compile. The Hex file is saved into the project directory as "projectname.Hex" which will be used by uiLAB Program utility to program the flash memory through the serial (RS232 or USB) interface. Below is the structure of the Hex file.



The file header is a 34 byte wide structure:

```
BYTE      hexCode1
BYTE      hexCode2
BYTE      hexCode3
BYTE      hexCode4
WORD      hexFileVersion
WORD      hexRes1
WORD      hexRes2
DWORD     hexCheckSum
DWORD     hexHeadersSize
DWORD     hexGUISize
DWORD     hexFontsSize
DWORD     hexBitmapsSize
DWORD     hexAudioSize
```

Where:
hexCode1 = 55
hexCode2 = 66
hexCode3 = 66
hexCode4 = 66
hexFileVersion = 100
hexRes1 and hexRes2 are reserved for future use.
hexCheckSum is file check sum as has been calculate by uiLab.

hexHeadersSize is the size of the headers section in bytes.
hexGUISize is the size of the GUI section in bytes.
hexFontsSize is the size of the fonts section in bytes.
hexBitmapsSize is the size of the bitmaps section in bytes.
hexAudioSize is the size of the audio section in bytes.

18. Flash Programming using the serial interface:

uiLAB has a utility to program the flash memory using the serial (RS232 or USB) interface. This utility will open the Hex file and place it in a large array. You can select which section or sections you need to program and then click on "program" and programming sequence will start. For example, if you change an object location or change the color of text then you only need to program the GUI section. If you modify the bitmaps list then you need to program Headers and Bitmaps sections.
Program utility use the program/erase terminal commands to program the flash memory. Each section must be erased first before programming. Programming is done by writing a page (256 byte) to the flash memory at the current page which can take some time to complete depending on the size of resources (fonts and bitmaps). The flash programming will not affect the user memory section. After programming is completed, you must reset or power OFF/ON SegeNT for the new configuration to take place.

19. Flash Programming from the micro SD card:

A very fast (~60 seconds) way to program the flash memory is to use micro SD card formatted to FAT16. The programming steps are:

1. Copy the Hex file from the project directory and paste it to you SD card and then rename it to "gui.hex"
2. Power OFF segeNT and set the dip-switches to Programming Mode.
3. Insert the SD card into SegeNT micro SD socket.
4. Power On segeNT.
5. A black background screen will appear and the SD card LED will turn ON if the card was successfully initialize and mounted by SegeNT.
6. If the file "gui.hex" was found and it is a valid hex file then, a countdown timer will appear on screen to warn you that programming will start in 10 seconds or less.
7. If you need to stop the programming sequence then, power SegeNT OFF and then remove the SD card.
8. If the 10 seconds time has elapsed then, a white background screen will appear and the programming sequence will start from the headers section down to the bitmaps section.
9. When programming is complete, power SegeNT Off, remove the SD card, set the dip-switches to Normal mode and then power SegeNT On.

20.TERMINAL COMMANDS:

Terminal commands are a set of serial commands that can be used by the host controller at any time to draw unsolicited graphic primitives and/or to control and monitor the SegeNT hardware. Terminal commands are different from the object commands in which they can be used at any time regardless if a screen is loaded and shown on the LCD or not. The below table lists the terminal command codes and functions:

Command Name	Code in decimal	Function
Get_ProductNumber	1	General Information
Get_FirmwareVersion	2	General Information
Get_SerialNumber	3	General Information
Get_UserVersion	4	General Information
Get_UserIDNumber0	5	General Information
Get_UserIDNumber1	6	General Information
Get_LCDSize	7	LCD
Get_TouchScreenType	8	LCD
Get_FlashMemorySize	9	Flash Memory
Get_UserFlashMemorySize	10	Flash Memory
Get_SystemStatus	11	System
Cal_TouchScreen	12	Touch Screen
Set_Brightness	13	LCD
Set_Backlight	14	LCD
Set_Beep	15	System
Set_LCD	16	LCD
Write_UserFlashMemory	17	Programming Flash Memory
Read_UserFlashMemory	18	Programming Flash Memory
Erase_UserFlashMemory	25	Programming Flash Memory
Set_DigitalOutput	19	System
Get_DigitalInput	20	System
Get_AnalogInput	21	System
Set_PWM	22	System
Get_TouchData	23	Touch Screen
Get_ProductCode	24	General Information
Play_Audio	26	Audio
Set_EntVolume	27	Audio
Set_ExtVolume	28	Audio
Mute	29	Audio
Clear_Screen	30	Drawing
Set_BackColor	31	Drawing
Set_DrawColor	32	Drawing
Set_DrawWidth	33	Drawing
Set_DrawType	34	Drawing
Draw_Line	35	Drawing
Draw_Rectangle	36	Drawing
Draw_FilledRectangle	37	Drawing
Draw_Bevel	38	Drawing
Draw_FilledBevel	39	Drawing
Draw_Circle	40	Drawing
Draw_FilledCircle	41	Drawing
Draw_Pixel	42	Drawing
Copy_Area	43	Drawing
Paste_Area	44	Drawing

Get_Pixel	45	Drawing
Set_ActiveLayer	46	Drawing
Get_ActiveLayer	47	Drawing
Get_CopyBuffer	48	Drawing
Draw_Bitmap	50	Bitmap
Show_Bitmap	51	Bitmap
Get_BitmapDimension	52	Bitmap
Get_BitmapBackStyle	53	Bitmap
Set_Font	60	Text
Write_Text	61	Text
Get_TextWidth	62	Text
Get_TextHeight	63	Text
Get_TextLength	64	Text
Get_TextLineCount	65	Text
Get_FontHeight	66	Text
Get_FontWidth	67	Text
Get_CurrentFont	68	Text
Get_FontColorDepth	69	Text
Program_Headers	70	Programming Flash Memory
Program_GUI	71	Programming Flash Memory
Program_Bitmaps	72	Programming Flash Memory
Program_Fonts	73	Programming Flash Memory
Program_Audio	79	Programming Flash Memory
Erase_FlashMemory	74	Programming Flash Memory
Erase_Headers	75	Programming Flash Memory
Erase_GUI	76	Programming Flash Memory
Erase_Bitmaps	77	Programming Flash Memory
Erase_Fonts	78	Programming Flash Memory
Erase_Audio	80	Programming Flash Memory

Below is the description of each command in details. For clarity, comma (,) is used to separate between the command bytes or the response bytes.

18.1. Get_ProductNumber

Description	Returns the 10 bytes (ASCII code) product number.
Command Code	1
Command	DVID, 0, 1, 1, CS
Response	ACK/NAK, 10, P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, CS
Arguments	P1-P10: product ASCII codes
Example	

18.2. Get_FirmWareVersion

Description	Returns the 5 bytes (ASCII code) version number
Command Code	2
Command	DVID, 0, 1, 2, CS
Response	ACK/NAK, 5, V1, V2, V3, V4, V5, CS
Arguments	V1-V5: firmware version ASCII codes
Example	

18.3. Get_SerialNumber

Description	Returns the 6 bytes (ASCII code) serial number
Command Code	3
Command	DVID, 0, 1, 3, CS
Response	ACK/NAK, 6, S1, S2, S3, S4, S5, S6, CS
Arguments	S1-S6: serial number ASCII codes
Example	

18.4. Get_UserVersion

Description	Returns the 2 bytes user defined version number. This number is set by the user and saved to the Configuration header in the flash memory.
Command Code	4
Command	DVID, 0, 1, 4, CS
Response	ACK/NAK, 2, UVH, UVL, CS
Arguments	UVH: user defined version number MSB UVL: user defined version number LSB
Example	

18.5. Get_UserIDNumber0

Description	Returns the 2 bytes user defined ID Number 0. This number is set by the user and saved to the Configuration header in the flash memory.
Command Code	5
Command	DVID, 0, 1, 5, CS
Response	ACK/NAK, 2, UID0H, UID0L, CS
Arguments	UID0H: user defined ID number 0 MSB UID0L: user defined ID number 0 LSB
Example	

18.6. Get_UserIDNumber1

Description	Returns the 2 bytes user defined ID Number 1. This number is set by the user and saved to the Configuration header in the flash memory.
Command Code	6
Command	DVID, 0, 1, 6, CS
Response	ACK/NAK, 2, UID1H, UID1L, CS
Arguments	UID1H: user defined ID number 1 MSB UID1L: user defined ID number 1 LSB
Example	

18.7. Get_LCDSize

Description	Returns the LCD width and height in pixels.
Command Code	7
Command	DVID, 0, 1, 7, CS
Response	ACK/NAK, 4, WidthH, WidthL, HeightL, HeightH, CS
Arguments	WidthH&WidthL: LCD width in pixels HeightH&HeightL: LCD height in pixels
Example	

18.8. Get_TouchScreenType

Description	Returns the touch screen type.
Command Code	8
Command	DVID, 0, 1, 8, CS
Response	ACK/NAK, 1, Type, CS
Arguments	Type = 0 => 4-wire resistive Type = 1 => 5-wire resistive Type = 2 => Capacitive Type = 3 => None
Example	

18.9. Get_FlashMemorySize

Description	Returns the size of the flash memory in MB.
Command Code	9
Command	DVID, 0, 1, 9, CS
Response	ACK/NAK, 1, Size, CS
Arguments	Size = 16 => 16MB Size = 32 => 32MB
Example	

18.10. Get_UserFlashMemorySize

Description	Returns the User flash memory size.
Command Code	10
Command	DVID, 0, 1, 10, CS
Response	ACK/NAK, 1, Size, CS
Arguments	Size = 0 => 64KB
Example	

18.11. Get_SystemStatus

Description	Returns the system status byte
Command Code	11
Command	DVID, 0, 1, 11, CS
Response	ACK/NAK, 1, Status, CS
Arguments	**Bit0, Bit1**: Mode 00 = Normal Mode 01 = Test Mode 10 = Touch screen Calibration Mode 11 = Programming Mode **Bit2, Bit3**: Not used. Read 0 **Bit4**: Touch Screen calibration status 0 = Calibration is done 1 = Calibration in progress **Bit5**: LCD status 0 = LCD is OFF 1 = LCD is ON **Bit6**: User Interface status 0 = User interface is disabled 1 = User interface is enabled **Bit7**: Power save status 0 = Power save is disabled 1 = Power save is enabled
Example	

18.12. Cal_TouchScreen

Description	Calibrate touch screen. Only valid for resistive type touch screen.
Command Code	12
Command	DVID, 0, 1, 12, CS
Response	ACK/NAK
Arguments	
Example	

18.13. Set_Brightness

Description	Set the LCD brightness level
Command Code	13
Command	DVID, 0, 2, 13, value, CS
Response	ACK/NAK
Arguments	Value: LCD brightness level from 0 to 127
Example	

18.14. Set_Backlight

Description	Enable or disable the LCD back light.
Command Code	14
Command	DVID, 0, 2, 14, value, CS
Response	ACK/NAK
Arguments	Value = 0 => backlight is OFF Value = 1 => backlight is ON
Example	

18.15. Set_Beep

Description	Turn the beeper on for a time specified by the Timer
Command Code	15
Command	DVID, 0, 2, 15, Timer, CS
Response	ACK/NAK
Arguments	Timer: is the duration of the beep in msec from 0 to 255
Example	

18.16. Set_LCD

Description	Enable or disable the LCD. This command will turn off the LCD timing signals and the back light. Frame buffer memory will not be changed so when you turn the LCD on again, the same graphics will be shown on the screen. This command can be used to save power during inactivity period.
Command Code	16
Command	DVID, 0, 2, 16, value, CS
Response	ACK/NAK
Arguments	Value = 0 => LCD is OFF Value = 1 => LCD is ON
Example	

18.17. Write_UserFlashMemory

Description	Write one page (256 bytes) to the user flash memory. Before writing any new pages, you need to send command "Erase_UserFlashMemory" first.
Command Code	17
Command	DVID, 1, 3, 17, PageH, PageL, B0, B1….B255,CS
Response	ACK/NAK
Arguments	PageH&PageL: Page number from 0 to 255 for 64KB user flash memory B0-B255: Page bytes Command NOB = 259 = 0x0103
Example	

18.18. Read_UserFlashMemory

Description	Read one page (256 bytes) from the user flash memory
Command Code	18
Command	DVID, 0, 3, 18, PageH, PageL, CS
Response	ACK/NAK, 255, B0, B1…. B255, CS
Arguments	PageH&PageL: Page number from 0 to 255 for 64KB user flash memory B0-B255: Page bytes
Example	

18.19. Erase_UserFlashMemory

Description	Erase one sector (4KB or 16 pages) of the user flash memory
Command Code	25
Command	DVID, 0, 2, 25, Sector, CS
Response	ACK/NAK
Arguments	Sector: Sector number from 0 to 15 for 64KB user flash memory
Example	

18.20. Set_DigitalOutput

Description	Set digital output value
Command Code	19
Command	DVID, 0, 3, 19, DO, Value, CS
Response	ACK/NAK
Arguments	DO: Digital output address from 0 to 7 Value: Digital output value 0 or 1
Example	

18.21. Get_DigitalInput

Description	Return the value of digital input
Command Code	20
Command	DVID, 0, 2, 20, DI, CS
Response	ACK/NAK, 1, Value, CS
Arguments	DI: Digital input address from 0 to 7 Value: Digital input value 0 or 1
Example	

18.22. Get_AnalogInput

Description	Return the value of analog input
Command Code	21
Command	DVID, 0, 2, 21, AI, CS
Response	ACK/NAK, 2, ValueH, ValueL CS
Arguments	AI: Analog input address from 0 to 7 ValueH&ValueL: Analog input value from 0 to 1023
Example	

18.23. Set_PWM

Description	Set PWM output
Command Code	22
Command	DVID, 0, 3, 22, PWM, ValueH, ValueL, CS
Response	ACK/NAK
Arguments	PWM: PWM output address from 0 to 1 ValueH&ValueL: PWM output value from 0 to 312
Example	

18.24. Get_TouchData

Description	Return the touch data
Command Code	23
Command	DVID, 0, 1, 23, CS
Response	ACK/NAK, 7, Status, XH, XL, YH, YL, Res1, Res2, CS
Arguments	Status: Pen Up/Down status. 1 = Pen Down, 0 = Pen Up XH&XL: Touch X coordinate YH&YL: Touch Y coordinate Res1, Res2: Reserved read 0
Example	

18.25. Get_ProductCode

Description	Return product code byte
Command Code	24
Command	DVID, 0, 1, 24, CS
Response	ACK/NAK, 1, Code, CS
Arguments	Code: Product code.
Example	

18.26. Play_Audio

Description	Play audio file from the flash memory or the built-in files
Command Code	26
Command	DVID, 0, 2, 26, ID, CS
Response	ACK/NAK
Arguments	ID: Audio file ID number. ID = 0 to 15: play audio files stored at the flash memory ID = 16 to 39: play a built-in sound
Example	

18.27. Set_EntVolume

Description	Set the built-in sound volume
Command Code	27
Command	DVID, 0, 2, 27, Vol, CS
Response	ACK/NAK
Arguments	Vol: Volume value from 0 to 100.
Example	

18.28. Set_ExtVolume

Description	Set the user defined audio volume
Command Code	28
Command	DVID, 0, 2, 28, Vol, CS
Response	ACK/NAK
Arguments	Vol: Volume value from 0 to 100.
Example	

18.29. Mute

Description	Turn the audio amplifier On or Off
Command Code	29
Command	DVID, 0, 2, 28, M, CS
Response	ACK/NAK
Arguments	M = 0 => OFF M = 1 => ON
Example	

18.30. Clear_Screen

Description	Fill the background with the default background color
Command Code	30
Command	DVID, 0, 1, 30, CS
Response	ACK/NAK
Arguments	
Example	

18.31. Set_BackColor

Description	Fill the screen with a Color
Command Code	31
Command	DVID, 0, 3, 31, ColorH, ColorL, CS
Response	ACK/NAK
Arguments	ColorH: Color high byte ColorL: Color low byte Color must be in RGB565 format
Example	Set the backcolor to White DVID, 0, 3, 31, 255, 255, CS

18.32. Set_DrawColor

Description	Set the Draw Color. This color will be used for drawing and writing.
Command Code	32
Command	DVID, 0, 3, 32, ColorH, ColorL, CS
Response	ACK/NAK
Arguments	ColorH: Color high byte ColorL: Color low byte Color must be in RGB565 format
Example	Set the draw color to Black DVID, 0, 3, 32, 0, 0, CS

18.33. Set_DrawWidth

Description	Set the draw width
Command Code	33
Command	DVID, 0, 2, 33, DW, CS
Response	ACK/NAK
Arguments	DW: Draw Width 0 = Normal (1 pixel) 1 = Thick (3 pixel)
Example	Set the draw width to Thick DVID, 0, 2, 33, 1, CS

18.34. Set_DrawType

Description	Set the Draw Type
Command Code	34
Command	DVID, 0, 2, 34, DT, CS
Response	ACK/NAK
Arguments	Value: Draw Type 0 = Solid 1 = Dash 2 = Dot
Example	Set the draw type to dash DVID, 0, 2, 34, 1, CS

18.35. Draw_Line

Description	Draw a line with the current Draw Color, Draw Width and Draw Type
Command Code	35
Command	DVID, 0, 9, 35, X1H, X1L, Y1H, Y1L, X2H, X2L, Y2H, Y2L, CS
Response	ACK/NAK
Arguments	X1H&X1L: X1 coordinate Y1H&Y1L: Y1 coordinate X2H&X2L: X2 coordinate Y2H&Y2L: Y2 coordinate
Example	Draw a line from (0,0) to (319,239) [Data in Hex] DVID, 0x00, 0x09, 0x23, 0x00, 0x00, 0x00, 0x00, 0x01, 0x3F, 0x00, 0xEF, CS

18.36. Draw_Rectangle

Description	Draw a rectangle with the current Draw Color, Draw Width and Draw Type
Command Code	35
Command	DVID, 0, 9, 35, X1H, X1L, Y1H, Y1L, X2H, X2L, Y2H, Y2L, CS
Response	ACK/NAK
Arguments	X1H&X1L: X1 coordinate Y1H&Y1L: Y1 coordinate X2H&X2L: X2 coordinate Y2H&Y2L: Y2 coordinate
Example	Draw a rectangle from (0,0) to (319,239) [Data in Hex] DVID, 0x00, 0x09, 0x24, 0x00, 0x00, 0x00, 0x00, 0x01, 0x3F, 0x00, 0xEF, CS

18.37. Draw_Bevel

Description	Draw a bevel (round rectangle) with the current Draw Color and Draw Width
Command Code	38
Command	DVID, 0, 10, 38, X1H, X1L, Y1H, Y1L, X2H, X2L, Y2H, Y2L, R, CS
Response	ACK/NAK
Arguments	X1H&X1L: X1 coordinate Y1H&Y1L: Y1 coordinate X2H&X2L: X2 coordinate Y2H&Y2L: Y2 coordinate R: Corner radius fixed to
Example	Draw a bevel from (0,0) to (64,64) with R = 30 [Data in Hex] DVID, 0x00, 0x0A, 0x26, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x40, 0x1E, CS

18.38. Draw_Circle

Description	Draw a circle with the current Draw Color and Draw Width
Command Code	40
Command	DVID, 0, 9, 40, X0H, X0L, Y0H, Y0L, RH, RL, 0, 0, CS
Response	ACK/NAK
Arguments	X0H&X0L: X0 coordinate Y0H&Y0L: Y0 coordinate RH&RL: Radius
Example	Draw a circle centered at (64,64) with a radius of 32 [Data in Hex] DVID, 0x00, 0x09, 0x28, 0x00, 0x40, 0x00, 0x40, 0x00, 0x20, 0x00, 0x00, CS

18.39. Draw_Pixel

Description	Draw a pixel with the current Draw Color
Command Code	42
Command	DVID, 0, 5, 42, XH, XL, YH, YL, CS
Response	ACK/NAK
Arguments	XH&XL: X coordinate YH&YL: Y coordinate
Example	Draw a pixel at (64,64) [Data in Hex] DVID, 0x00, 0x05, 0x2A, 0x00, 0x40, 0x00, 0x40, CS

18.40. Draw_FilledBevel

Description	Draw a filled bevel (round rectangle) with the current Draw Color
Command Code	39
Command	DVID, 0, 10, 39, X1H, X1L, Y1H, Y1L, X2H, X2L, Y2H, Y2L, R, CS
Response	ACK/NAK
Arguments	X1H&X1L: X1 coordinate Y1H&Y1L: Y1 coordinate X2H&X2L: X2 coordinate Y2H&Y2L: Y2 coordinate R: Corner radius
Example	Draw a filled bevel from (0,0) to (64,64) with R = 30 [Data in Hex] DVID, 0x00, 0x0A, 0x27, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x40, 0x1E, CS

18.41. Draw_FilledCircle

Description	Draw a filled circle with the current Draw Color
Command Code	41
Command	DVID, 0, 9, 41, X0H, X0L, Y0H, Y0L, RH, RL, 0, 0, CS
Response	ACK/NAK
Arguments	X0H&X0L: X0 coordinate Y0H&Y0L: Y0 coordinate RH&RL: Radius
Example	Draw a filled circle centered at (64,64) with a radius of 32 [Data in Hex] DVID, 0x00, 0x09, 0x29, 0x00, 0x40, 0x00, 0x40, 0x00, 0x20, 0x00, 0x00, CS

18.42. Draw_FilledRectangle

Description	Draw a filled rectangle with the current Draw Color
Command Code	36
Command	DVID, 0, 9, 36, X1H, X1L, Y1H, Y1L, X2H, X2L, Y2H, Y2L, CS
Response	ACK/NAK
Arguments	X1H&X1L: X1 coordinate Y1H&Y1L: Y1 coordinate X2H&X2L: X2 coordinate Y2H&Y2L: Y2 coordinate
Example	Draw a rectangle from (0,0) to (319,239) [Data in Hex] DVID, 0x00, 0x09, 0x24, 0x00, 0x00, 0x00, 0x00, 0x01, 0x3F, 0x00, 0xEF, CS

18.43. Copy_Area

Description	Copy a screen area to internal buffer. The internal buffer is 1024 pixels (2Kbyte) wide. The copied number of pixels should not exceed 1024 pixels.
Command Code	43
Command	DVID, 0, 7, 43, XH, XL, YH, YL, W, H, CS
Response	ACK/NAK
Arguments	XH&XL: Area left top corner X coordinate YH&YL: Area left top corner Y coordinate W: Area Width H: Area Height Note: W * H <= 1024
Example	Copy a screen area at (0,0) with W = 32 and H = 32 DVID, 0x00, 0x07, 0x2B, 0x00, 0x00, 0x00, 0x00, 0x20, 0x20, CS

18.44. Paste_Area

Description	Paste an area from the internal buffer. The internal buffer is 1024 pixels (2Kbyte) wide The pasted number of pixels should not exceed 1024 pixels.
Command Code	44
Command	DVID, 0, 7, 44, XH, XL, YH, YL, W, H, CS
Response	ACK/NAK
Arguments	XH&XL: Area left top corner X coordinate YH&YL: Area left top corner Y coordinate W: Area Width H: Area Height Note: W * H <= 1024
Example	Paste an area from the internal buffer at (0,0) with W = 32 and H = 32 DVID, 0x00, 0x07, 0x2C, 0x00, 0x00, 0x00, 0x00, 0x20, 0x20, CS

18.45. Get_Pixel

Description	Return Pixel color
Command Code	45
Command	DVID, 0, 5, 44, XH, XL, YH, YL, CS
Response	ACK/NAK, 2, ColorH, ColorL, CS
Arguments	XH&XL: Pixel X coordinate YH&YL: Pixel Y coordinate ColorH&ColorL: Pixel Color
Example	

18.46. Set_ActiveLayer

Description	Select the active layer. All drawing, bitmaps and graphics will be performed on this layer
Command Code	46
Command	DVID, 0, 2, 46, Layer, CS
Response	ACK/NAK
Arguments	Layer = 0 => Main Layer = 1 => PIP1 Layer = 2 => PIP2
Example	

18.47. Get_ActiveLayer

Description	Return the active layer
Command Code	47
Command	DVID, 0, 1, 47, CS
Response	ACK/NAK, 1, Layer, CS
Arguments	Layer = 0 => Main Layer = 1 => PIP1 Layer = 2 => PIP2
Example	

18.48. Get_CopyBuffer

Description	Read one block (256 bytes or128 pixels) of the copy buffer
Command Code	48
Command	DVID, 0, 2, 47, Block, CS
Response	ACK/NAK, 255, P0H, P0L, P1H, P1L…P127H, P127L, CS
Arguments	Block: Block Number from 0 to 7. PxH&PxL: Pixel Color in RGB565 format Note: To fully read the whole buffer (1024 pixels), you need to send this command 8 times with block = 0 first then block = 1 and so on.
Example	

18.49. Draw_Bitmap

Description	Draw a bitmap directly from the serial interface. The bitmaps will be drawn block by block and each block is 256 bytes or 128 pixels wide. The minimum bitmap size (2*W*H) is 256 bytes (one block) and maximum is screen size (2*LCDX*LCDY).
Command Code	50
Command	DVID, 1, 11, 50, XH, XL, YH, YL, WidthH, WidthL, HeightH, HeightL, BlockH, BlockL, P0H, P0L, P1H, P1L…P127H, P127L, CS
Response	ACK/NAK
Arguments	XH&XL: Left YH&YL: Top WidthH&WidthL: Bitmap Width in pixels HeightH&HeightL: Bitmap Height in pixels BlockH&BlockL: Block Number PxH&PxL: Pixel color in RGB565 format
Example	

18.50. Show_Bitmap

Description	Draw a bitmap that already saved in the flash memory
Command Code	51
Command	DVID, 0, 9, 51, IDH, IDL, XH, XL,YH, YL, Orientation, Mirror, CS
Response	ACK/NAK
Arguments	IDH&IDL: Bitmap ID from 0 to 511 XH&XL: Left YH&YL: Top Orientation: Bitmap rotation 0 = 0º 1 = 90º 2 = 180º 3 = 270º Mirror: Mirror the bitmap 0 = No 1 = Yes Please note that orientation and mirror only available for opaque back style bitmaps and not always give a good results.
Example	

18.51. Get_BitmapDimension

Description	Return the bitmap width and height. The bitmap must be already saved into the flash memory.
Command Code	52
Command	DVID, 0, 3, 52, IDH, IDL, CS
Response	ACK/NAK, 4, WidthH, WidthL, HeightH, HeightL, CS
Arguments	IDH&IDL: Bitmap ID from 0 to 511 WidthH&WidthL: Bitmap Width in Pixels HeightH&HeightL: Bitmap Height in Pixels
Example	

18.52. Get_BitmapBackStyle

Description	Return the bitmap back style. The bitmap must be already saved into the flash memory
Command Code	53
Command	DVID, 0, 3, 53, IDH, IDL, CS
Response	ACK/NAK, 1, BackStyle, CS
Arguments	IDH&IDL: Bitmap ID from 0 to 511 BackStyle: Bitmap Back Style 0 = Opaque 1 = Transparent
Example	

18.53. Set_Font

Description	Set the current active font. Text will be written using this font. Default font is Font0
Command Code	60
Command	DVID, 0, 2, 60, ID, CS
Response	ACK/NAK
Arguments	ID: Font ID from 0 to 7
Example	Set the current font to Font7 (ID = 7) DVID, 0, 2, 60, 7, CS

18.54. Write_Text

Description	Write text with the current font, draw color and at CurX & CurY
Command Code	61
Command	DVID, NOBH, NOBL, 61, CurXH, CurXL, CurYH, CurYL, <String>, CS
Response	ACK/NAK
Arguments	NOBH&NOBL = 5 + NOC, where NOC is the String Length (number of characters) including the termination code "NULL" CurXH&CurXL: Cursor X coordinate CurYH&CurYL: Cursor Y coordinate <String>: Up to 256 ASCII character codes including the termination code (NULL). To start with a new line, insert new line code (0x0A). Note: The string must be terminated by NULL = 0x00.
Example	Write "HELLO" at (0,0) DVID, 0x00, 0x0B, 0x3D, 0x00, 0x00, 0x00, 0x00, 0x48, 0x45, 0x4C, 0x4C, 0x4F, 0x00, CS

18.55. Get_TextWidth

Description	Return text width. If the text is a multi line, then the widest line will be returned.
Command Code	62
Command	DVID, NOBH, NOBL, 62, <String>, CS
Response	ACK/NAK, 2, WH, WL, CS
Arguments	NOBH&NOBL = 1 + NOC, where NOC is the String Length (number of characters) including the termination code "NULL" <String>: Up to 256 ASCII character codes including the termination code (NULL). WH&WL: Text Width in Pixels. If the text has more than one line, the widest line will be returned. Note: The string must be terminated by NULL = 0x00.
Example	

18.56. Get_TextHeight

Description	Return text height.
Command Code	63
Command	DVID, NOBH, NOBL, 63, <String>, CS
Response	ACK/NAK, 2, HH, HL, CS
Arguments	NOBH&NOBL = 1 + NOC, where NOC is the String Length (number of characters) including the termination code "NULL" <String>: Up to 256 ASCII character codes including the termination code (NULL). HH&HL: Text Height in Pixels Note: The string must be terminated by NULL = 0x00.
Example	

18.57. Get_TextLength

Description	Return text length (text number of characters)
Command Code	64
Command	DVID, NOBH, NOBL, 64, <String>, CS
Response	ACK/NAK, 2, LH, LL, CS
Arguments	NOBH&NOBL = 1 + NOC, where NOC is the String Length (number of characters) including the termination code "NULL" <String>: Up to 256 ASCII character codes including the termination code (NULL). LH&LL: Text Length in characters Note: The string must be terminated by NULL = 0x00.
Example	

18.58. Get_TextLineCount

Description	Return text number of lines.
Command Code	65
Command	DVID, NOBH, NOBL, 65, <String>, CS
Response	ACK/NAK, 2, LCH, LCL, CS
Arguments	NOBH&NOBL = 1 + NOC, where NOC is the String Length (number of characters) including the termination code "NULL" <String>: Up to 256 ASCII character codes including the termination code (NULL). LCH&LCL: Text number of lines Note: The string must be terminated by NULL = 0x00.
Example	

18.59. Get_FontHeight

Description	Return Font height.
Command Code	66
Command	DVID, 0, 2, 66, ID, CS
Response	ACK/NAK, 1, FH, CS
Arguments	ID: Font ID from 0 to 7 FH: Font height in pixels
Example	

18.60. Get_FontWidth

Description	Return the width of each character in the font.
Command Code	67
Command	DVID, 0, 2, 67, ID, CS
Response	ACK/NAK, 224, W0, W1...W223, CS
Arguments	ID: Font ID from 0 to 7 W0-W223: is the width of the character starting from character 0 (ASCII 32) to character 223 (ASCII 255).
Example	

18.61. Get_Font

Description	Return the current (active) font.
Command Code	68
Command	DVID, 0, 1, 68, CS
Response	ACK/NAK, 1, ID, CS
Arguments	ID: Font ID from 0 to 7
Example	

18.62. Get_FontColorDepth

Description	Return the Font ColorDepth.
Command Code	69
Command	DVID, 0, 2, 69, ID, CS
Response	ACK/NAK, 1, FCD, CS
Arguments	ID: Font ID from 0 to 7 FCD = 0 => 1BPP Color Depth FCF = 1 => 2BPP Color Depth
Example	

18.63. Program_Headers

Description	Write one page (256 bytes) to the Headers section of the flash memory. Before writing any new pages, you need to send command “Erase_Headers” first.
Command Code	70
Command	DVID, 1, 3, 70, PageH, PageL, B0, B1….B255, CS
Response	ACK/NAK
Arguments	PageH&PageL: Page number from 0 to 33 B0-B255: Page bytes Command NOB = 259 = 0x0103
Example	

18.64. Program_GUI

Description	Write one page (256 bytes) to the GUI section of the flash memory. Before writing any new pages, you need to send command “Erase_GUI” first.
Command Code	71
Command	DVID, 1, 3, 71, PageH, PageL, B0, B1….B255, CS
Response	ACK/NAK
Arguments	PageH&PageL: Page number from 0 to (GUI_Size – 1) in pages B0-B255: Page bytes Command NOB = 259 = 0x0103
Example	

18.65. Program_Bitmaps

Description	Write one page (256 bytes) to the Bitmaps section of the flash memory. Before writing any new pages, you need to send command “Erase_Bitmaps” first.
Command Code	72
Command	DVID, 1, 3, 72, PageH, PageL, B0, B1….B255, CS
Response	ACK/NAK

Arguments	PageH&PageL: Page number from 0 to (Bitmaps_Size – 1) in pages B0-B255: Page bytes Command NOB = 259 = 0x0103
Example	

18.66. Program_Fonts

Description	Write one page (256 bytes) to the Fonts section of the flash memory. Before writing any new pages, you need to send command "Erase_Fonts" first.
Command Code	73
Command	DVID, 1, 3, 73, PageH, PageL, B0, B1….B255, CS
Response	ACK/NAK
Arguments	PageH&PageL: Page number from 0 to (Fonts_Size – 1) in pages B0-B255: Page bytes Command NOB = 259 = 0x0103
Example	

18.67. Program_Audio

Description	Write one page (256 bytes) to the Audio section of the flash memory. Before writing any new pages, you need to send command "Erase_Audio" first.
Command Code	79
Command	DVID, 1, 3, 79, PageH, PageL, B0, B1….B255, CS
Response	ACK/NAK
Arguments	PageH&PageL: Page number from 0 to (Audio_Size – 1) in pages B0-B255: Page bytes Command NOB = 259 = 0x0103
Example	

18.68. Erase_FlashMemory

Description	Erase the whole flash memory.
Command Code	74
Command	DVID, 0, 3, 74, C1, C2, CS
Response	ACK/NAK
Arguments	C1 = 0x55 C2 = 0xAA Erasing the whole flash memory will take some time to be done. It is not recommended to use this command.
Example	

18.69. Erase_Headers

Description	Erase the Headers section of the flash memory.
Command Code	75
Command	DVID, 0, 3, 75, C1, C2, CS
Response	ACK/NAK
Arguments	C1 = 0x55 C2 = 0xAA
Example	

18.70. Erase_GUI

Description	Erase the GUI section of the flash memory.
Command Code	76
Command	DVID, 0, 3, 76, C1, C2, CS
Response	ACK/NAK
Arguments	C1 = 0x55 C2 = 0xAA
Example	

18.71. Erase_Fonts

Description	Erase the Fonts section of the flash memory.
Command Code	77
Command	DVID, 0, 3, 77, C1, C2, CS
Response	ACK/NAK
Arguments	C1 = 0x55 C2 = 0xAA
Example	

18.72. Erase_Bitmaps

Description	Erase the Bitmaps section of the flash memory.
Command Code	78
Command	DVID, 0, 3, 78, C1, C2, CS
Response	ACK/NAK
Arguments	C1 = 0x55 C2 = 0xAA
Example	

18.73. Erase_Audio

Description	Erase the Audio section of the flash memory.
Command Code	80
Command	DVID, 0, 3, 80, C1, C2, CS
Response	ACK/NAK
Arguments	C1 = 0x55 C2 = 0xAA
Example	

19.Graphical User Interface (GUI):

"Desktop software engineers never have to write code for line drawing or even for a button, so why should embedded engineers?" Abdul Haidar, CEO

In most desktop programming languages, a control like a Button is an object and has, for example, a caption property and a click event to which you can attach a method. It acts as a black-box, as you the programmer do not have to worry how the button responds to the click input or how it looks – that has been sorted out for you.
At Haidar Technology, we believe that embedded engineers should have the same capability that desktop developers have when it comes to embedded GUI. Our solution is a combination of Serial- Enabled GUI Engine (Sege) and a true WYS|WYG drag-and-drop Visual GUI builder "uiLAB" that allows the embedded engineer to design the GUI by dragging the objects onto a simulated screen and then altering their properties. When the designer is happy with the layout, the tool will automatically generate code (Hex file) for the target hardware.
If the layout or some bitmaps need to be redesigned just disconnects the target hardware from the controlling processor (host) and moves it to a serial port of a PC. Program the new code and you have the new user interface in place! Pure visual changes do not require modifications to the host controller code!
Since "SEGE" interacts with the host controller through a serial protocol, integration with a specific processor or platform, RTOS, or compiler is not required. Also, the requirements for the host controller (memory and speed) are significantly less!
"Sege" contains a set of powerful pre-coded objects. Each object has a set of properties, methods and events that define the behavior of the object and how will be manipulated by the host controller. The designer only needs to know how to input data to the object to get certain results out but does not need to know what goes inside the object. The host controller interacts with the GUI via a set of serial commands called "GUI Commands".

19.1 Object Code:

Each object has a unique code, which specifies the type of the object by its class name.
The table below lists the objects and their codes:

Object	Code in Dec	Description
TextBox	1	Displays a dynamic or static text
Button	2	Displays a push or momentary button.
Needle	3	Displays a radial gauge.
NumberBox	4	Displays bitmap style number.
Slider	5	Displays horizontal or vertical slider.
Chart	6	Displays YT or XY line graph.
BarGraph	7	Displays horizontal or vertical progress bar or bar graph.
PictureBox	8	An area of the screen that can be used for shape drawing, writing dynamic text, viewing pictures, hand drawing and a lot more.
Image	9	Displays a single bitmap or an array of bitmaps. It also displays GIF animation.
Shape	10	Displays shape like line, rectangle, circle …
Form	11	It is a control or object container, which can hold one or more objects. Form can be Screen, Window1 or Window2.

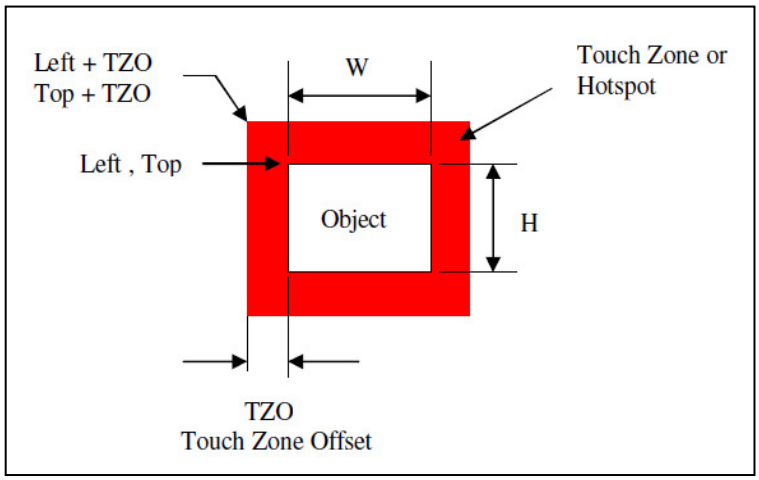
19.2 Object ID

At design time, each object will have a unique number called "ID". The Object ID number is automatically generated by "uiLAB" and cannot be altered at run time. The Object ID number will be used by the host to interact with specific object on specific form.
Each form in the GUI project can hold up to 64 objects and there is an offset of 64 in the object ID number based on the type of the form:

Form Type	Object ID Number Range
Screen	From 0 to 63
Window1	From 64 to 127
Window2	From 128 to 191

19.3 Input Interface (Touch Event):

All objects have a "TouchZone" property, when it is enabled, an invisible touch zone or hotspot will be associated with the object. The default size of the touch zone is the same as the object but it can be little bigger than the object by altering "TZOffset" property. A touch event will be generated when an object is being touched.



19.4 Object Touch Functions or Events:

Touch functions are built-in functions that will be called when a touch event is generated. Each object has its own set of touch functions which can be enabled or disabled by the host. Below is a list of all object functions:

<table>
<tr><th>Function or Event</th><th>Description</th></tr>
<tr><td>Sound</td><td>Generate a sound effect when the object is touched. Any of the built-in sound effects can be used to generate a click sound or simple buzzer sound.
This function is available for all objects.</td></tr>
<tr><td>Notify Host</td><td>Drive NotifyHost hardware pin Active (Low) for ~ 10 msec. This will interrupt the host controller and notify it that an object has been touched and ready to be serviced. The host then will read the "uiMessage" and then responses to the user input. This minimizes the frequency of interruptions to the normal work of the host.
This event is available for all objects.
NotifyHost event can be set to the following modes:
<table>
<tr><th>Notify Host Mode</th><th>NotifyHost hardware Pin</th></tr>
<tr><td>None</td><td>Inactive</td></tr>
<tr><td>OnPress</td><td>Active when the object has been touched.</td></tr>
<tr><td>OnChange</td><td>Active when the object value has been changed.</td></tr>
<tr><td>OnPressAndHold</td><td>Active as long as the button is being pressed.</td></tr>
<tr><td>OnRelease</td><td>Active when pen is up.</td></tr>
</table>
Please see each object description for more information.</td></tr>
<tr><td>Toggle State</td><td>Toggle the button state. If the button is disabled (State = 2) then, it will ignore the Touch Event. Only valid for Button.</td></tr>
<tr><td>Move Thumb</td><td>Move the slider thumb. Only valid for Slider.</td></tr>
<tr><td>Sketch</td><td>Enable PictureBox hand drawing. Only valid for PictureBox.</td></tr>
<tr><td>Special</td><td>Only available for TextBox and Image objects.</td></tr>
</table>

19.5 User Interface Message (uiMessage):

The uiMessage is updated at a rate defined by "uiUpdateRate" value in the Configuration header. The default value for "uiUpdateRate" is 100 msec and can be set to 200, 300, 400 or 500 msec. Every uiUpteRate value, SegeNT uiService routine will scan the visible objects with enabled "TouchZone" to see if any object is being or has been touched. If yes, TouchEvent will be generated and uiMessage will be populated with the object values.
In order to design a responsive GUI with no missing events, the host needs to read uiMessage as soon as possible and this can be done by enabling "NotifyHost" event for every object with enabled "TouchZone". In this case, NotifyHost pin will interrupt the host at the falling edge, the host then will send "Get_uiMessage" command to read the uiMessage.
Below is the uiMessage structure:

Variable	Type	Description
Status	BYTE	User Interface Status. 0 = Invalid Event 1 = Valid Event
Type	BYTE	Touch Event Type. 1 = Press 2 = Release 3 = Move or Hold
Code	BYTE	Object Code. The code of the object who had received the touch event.
ID	BYTE	Object ID. The ID of the object who had received the touch event.
Value	WORD	Object Value. Only Valid for Buttons and Sliders. The Value of the object who had received the touch event.

19.6 Object Properties:

The data inside an object is known as the **properties** of the object. In familiar terms, it corresponds to the values of variables, which are stored inside the object. For example, all objects have IDs and it is a property. Object properties are just like variable in which they can have a type. They can be Byte, Word, DWord, or String and can be retrieved and set using "Get_ObjProperty "and "Set_Obj Property" GUI commands. Not all properties can be set or changed by the host. Those properties are set at design time and cannot be changed at run time.
Each property has a unique number called "Property Index". This number is used by the host to access the property.
Each object has its own set of properties. The table below lists the common properties among all objects:

Property Name	Index	Type	Access	Description
Code	0	Byte	-	Object code
ID	1	Byte	-	Object ID
Left	2	Word	R	Object X position relative to the form left edge
Top	4	Word	R	Object Y position relative to the form top edge
Width	6	Word	R	Object width
Height	8	Word	R	Object height

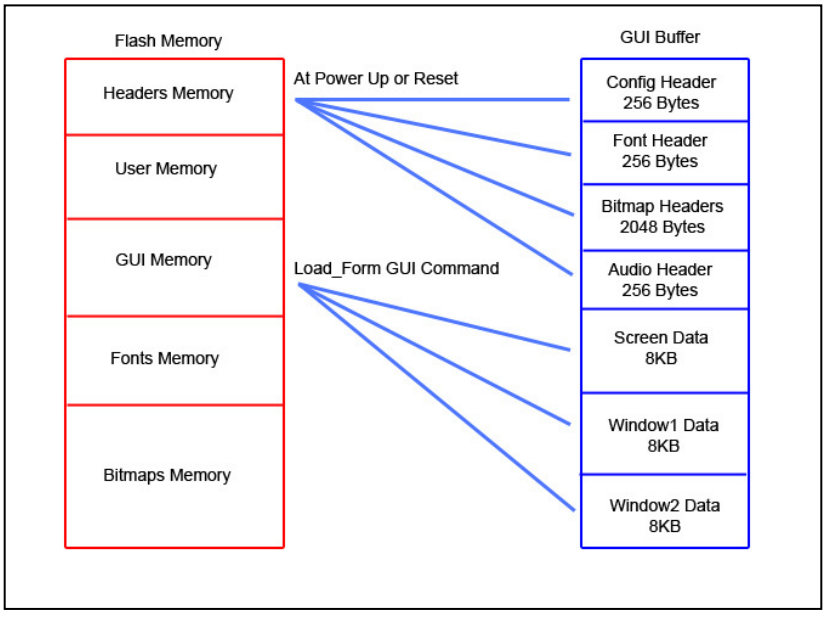
19.7 Object Standard Colors:

Standard colors are user defined colors used to draw the object 3D border and other standard style objects. They are saved to the Configuration Header in RGB565 format.
You can edit them from "uiLab" (Edit -> Object Colors).

Object Standard Color	Description	Default RGB888Value
3D Dark	3D Border Dark Color	RGB (172, 168, 153)
3D Darkest	3D Border Darkest Color	RGB (113, 111, 100)
3D Light	3D Border Light Color	RGB (241, 239, 226)
3D Lightest	3D Border Lightest Color	RGB (255, 255, 255)
Active Button Face	Enabled Button Face Color	RGB (236, 233, 216)
Disable Button Face	Disabled Button Face Color	RGB (236, 233, 216)
Disable Text	Disabled Text Color	RGB (172, 168, 153)
HighLight	Highlighted Object Color	RGB (49, 106, 197)
HighLight Text	Highlighted Object Text Color	RGB (255, 255, 255)
Active Window Frame	Enabled Window Frame Color	RGB (0, 84, 227)
Disable Window Frame	Disabled Window Frame Color	RGB (212, 208, 200)
Window Background	Form Back Ground Color	RGB (255, 255, 255)

19.8 GUI Buffer:

The GUI Buffer is an area of the main controller internal RAM assigned for the GUI data and headers. At power up or after a reset, the headers data is moved from the flash memory to the GUI Header Section. When the host sends "Load_Form" GUI command, Form data will be moved from the flash to the GUI Form Section. Based on the Form type (screen, window1 or window2), data will be saved to corresponding area of the From Section. Three Forms (one screen, one window1 and one window2) can be saved to the GUI Buffer at a time.



19.9 Object Draw Priority:

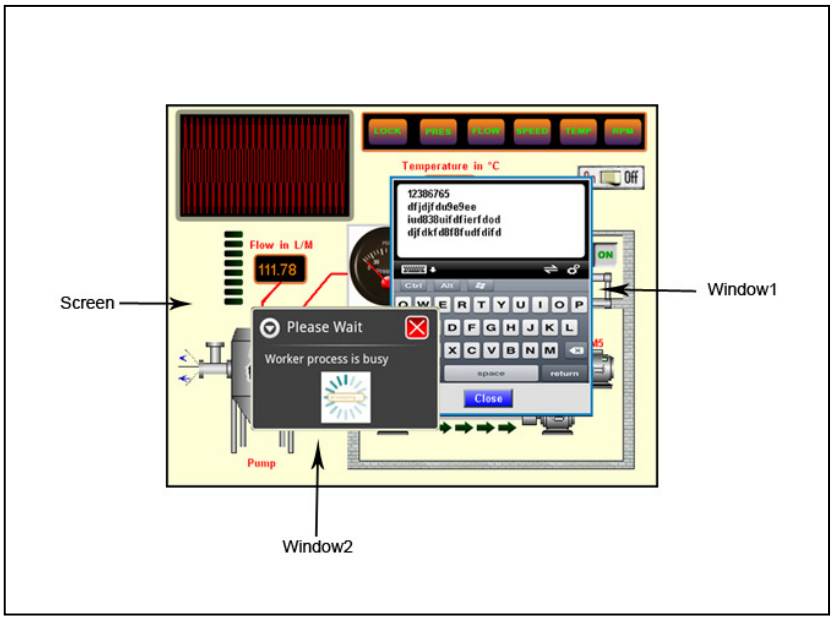
Object Draw Priority is basically the order in which objects are drawn on a form. After sending the GUI command “Form_Draw”, objects are drawn according to the following order:
Form, Shapes, Picture Boxes, Images, Buttons, Bar Graphs, Text Boxes, Number Boxes, Sliders, Charts, and last Needles.
This allows you to use shapes and picture boxes as container for another object like buttons, sliders ….

19.10 Object UI Priority:

Object UI Priority is basically the order in which objects are scanned by the User Interface routine to check if an object is being touched by the user. Objects with enabled “TouchZone” property are scanned according to the following order:
Buttons, Sliders, Images, Text Boxes, Picture Boxes, Number Boxes, Shapes, Charts, Bar Graphs and last Needles.
For example, if you have a button over a shape and both have “TouchZone” enabled, if you press on the shape only, the shape will generate a touch event, but when you press on the button, then only the button will generate the touch event.

20. Graphical User Interface Objects:

20.1. Form:



Form is basically an object container. Form has three different types:

- Screen: has the same width and height as the target display. It is not movable and always drawn at (0, 0). Every GUI project must have at least one screen.
- Window1: sizable and movable. The maximum size of Window1 depends on the target display resolution. The host can view or hide it without modifying the background image.
- Window2: sizable and movable. The maximum size of Window2 depends on the target display resolution. The host can view or hide it without modifying the background image.

SegeNT is capable of displaying up to three forms (one screen, one window1 and one window2) at the same time. Each form has its frame buffer and GUI buffer. When a form is hidden (not shown on the display), its pixels are removed from the display but they are still in the frame buffer so the host can still modify the hidden form and when it is viewed again, the new pixels will appear on the display.
Alpha blending between the screen and window1&2 or just between window1 and window2 is possible. Please see "Special Effects" for more information.
The form User Interface (ui) can be enabled or disabled by the host. Enabling the form user interface means that the form objects will generate touch events. One visible form (screen, window1 or window2) will generate touch events (accept user touch) at a time. For example, if a screen, window1 and window2 are visible on the display and all are enabled, then window2 only will accept user touch, if window2 is hidden, then window1 will accept user touch, if window1 is hidden then, the screen will accept the user touch.

Form Properties

Name	Index	Type	Access	Description
Type	10	Byte	R	Form type. 0 = Screen 1 = Window1 2 = Window2
Style	11	Byte	R/W	Background style 0 = Solid background color 1 = Background bitmap
BackStyle	12	Byte	R/W	Back Style

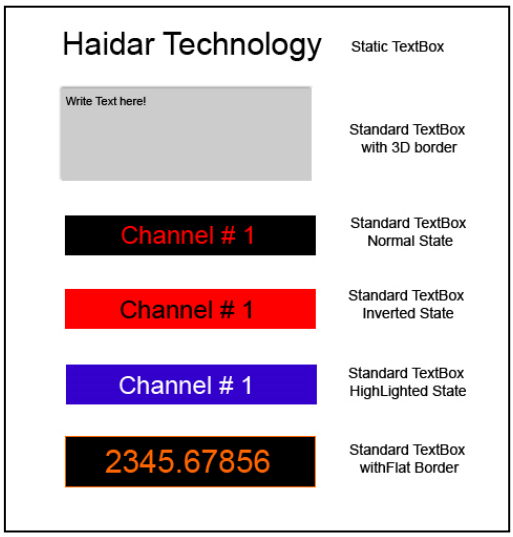
				0 = Opaque 1 = Transparent
BorderStyle	13	Byte	R/W	Border Style 0 = None 1 = Thick line
uiMode	14	Byte	R/W	Enable/Disable form user interface 0 = Disabled 1 = Enabled
BGBitmapID	16	Word	R/W	Background bitmap ID
BackColor	18	Word	R/W	Background color in RGB565 format
BorderColor	20	Word	R/W	Border color in RGB565 format
TransparentColor	22	Word	R/W	Transparent color in RGB565 format
OffsetX	24	Byte	R	Read 0
OffsetY	75	Byte	R	Read 0

Form GUI Commands

See "GUI Commands" For more Information.

GUI Command	**Description**
Form_Load	Load the form data form the flash memory to the GUI memory
Form_Draw	Show the form and its objects on the target display
Form_View	View or Hide the form
Form_Move	Move the form to new position
Form_Enable	Enable/Disable the form user interface
Form_GetStatus	Return the form user interface status (uiMode property)

20.2. TextBox:



The textbox is an area of the display that allows the host to enter and edit text. The maximum number of characters that can be stored in the textbox object at any time is 64 characters including the terminal character "NULL". There are two styles of the TextBox:

- **Standard**: text can be changed or modified by the host.
- **Static**: text is static and cannot be changed by the host. This style is used to print text directly to the form without the background rectangle or the border.

The text saved in the "Text" property will be displayed when the screen is initially loaded and shown. The text will be clipped if its width or height higher than the textbox width or height.
The state of a standard textbox can be set by the host or by the user "touch event" to Normal, Highlight or Invert:

- Normal: Fore color is used for text and the Back color is used for the background rectangle.
- Highlight: Object color "Highlight Text" is used for the text and object color "Highlight" is used for the background color.
- Invert: The opposite of the normal state.

Properties

Name	Index	Type	Access	Description
Style	10	Byte	R	TextBox Style 0 = Standard 1 = Static
Alignment	11	Byte	R/W	Text alignment 0 = Left 1 = Center 2 = Right
Font	12	Byte	R/W	Font from 0 to 7
BorderStyle	13	Byte	R	Border Style 0 = None 1 = Flat 2 = 3D
BackColor	14	Word	R/W	Back color in RGB565 format
ForeColor	16	Word	R/W	Fore color in RGB565 format
BorderColor	18	Word	R/W	Border color in RGB565 format
OffsetX	20	Byte	R	Width offset
OffsetY	21	Byte	R	Height offset

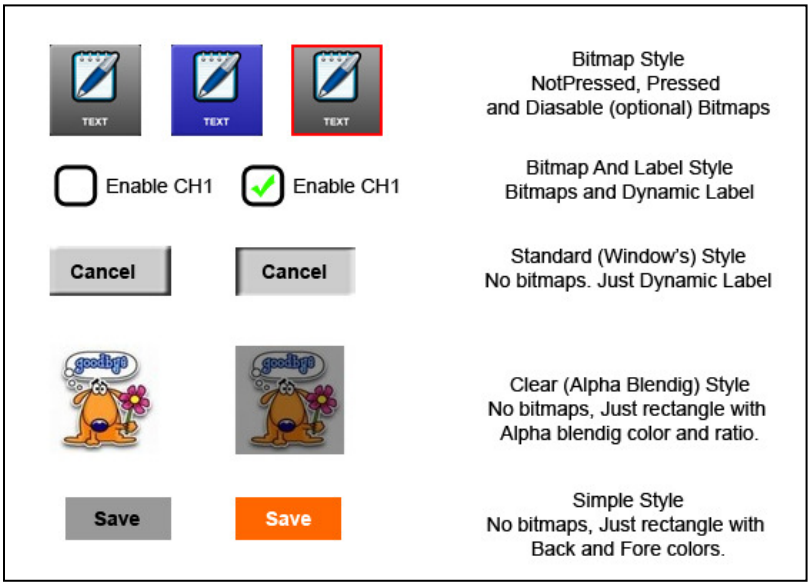
TouchZone	22	Byte	R	Enable/Disable Touch zone 0 = Disabled 1 = Enabled
TouchZoneOffset	23	Byte	R/W	Touch Zone Offset
SoundEvent	24	Byte	R/W	Enable/Disable Sound Event 0 = Disabled 1 = Enabled
NotifyHostEvent	25	Byte	R/W	Enable/Disable Notify Host Event 0 = Disabled 1 = Enabled (OnPress)
SpecialEvent	36	Byte	R/W	Enable/Disable TextBox Special Event 0 = Disabled 1 = Highlight 2 = Invert
State	35	Byte	R/W	TextBox State 0 = Normal 1 = Highlighted 2 = Inverted
BeepDuration	37	Byte	R/W	Sound Period in msec
Text	64	String	W	Textbox Text. Up to 64 ASCII characters including the termination code “NULL”
FunCode	26	Byte	R/W	Function Code. Not used.
FunVal0	27	Byte	R/W	Function Value 0. Not used.
FunVal1	28	Byte	R/W	Function Value 1. Not used.
FunVal2	29	Byte	R/W	Function Value 2. Not used.
FunVal3	30	Byte	R/W	Function Value 3. Not used.
FunVal4	31	Byte	R/W	Function Value 4. Not used.
FunVal5	32	Byte	R/W	Function Value 5. Not used.
FunVal6	33	Byte	R/W	Function Value 6. Not used.
FunVal7	34	Byte	R/W	Function Value 7. Not used.

TextBox GUI Commands

See “GUI Commands” For more Information.

GUI Command	**Description**
TextBox_Clear	Clear textbox text
TextBox_Text	Write Text to a textbox
TextBox_State	Set textbox State
TextBox_Number	Write a number (signed long) to a textbox

20.3. Button:



Button is the most commonly used object in any GUI design. Button is basically an area of the display that reacts to the user touch. The host can set any button to Disable state in which the button will not generate any Touch event until is enabled again.

The button object has 5 different styles:

- **Bitmap**: the button is made form 3 bitmaps, one for NotPressed state, the second for Pressed state and third one is for Disable state (optional).
- **BitmapAndLabel**: is the same as Bitmap style but it also has a label that can be modified by the host. In disable state, the label color will be changed to Object Color "Disable Text".
- **Standard**: is as same as Window's style button. No bitmaps are needed for this style. Label can be modified by the host. The colors used to draw the button are saved in the Configuration header under "Object Colors". . In disable state, the label color will be changed to Object Color "Disable Text".
- **Clear**: is basically a clear rectangle with Alpha blending color and ratio. It can be placed over any part of the background image to convert it into a button.
- **Simple**: it is the fastest to draw button style. Label and Inactive back colors are used to draw the NotPressed state while Active label and Active back colors are used to draw the pressed state. Border is also available for this style only. In disable state, the label color will be changed to Object Color "Disable Text".

Button also has two types:

- **Momentary**: it is the same as clickable button. The button state will go from NotPressed to Pressed when it is pressed and from Pressed to Notpressed again when it is released. Touch Event for this type is always generated when the button is released.
- **Latching**: this type maintains its state after being touched. The button state will go from ON to OFF or from OFF to ON when it is pressed. Touch Event for this type is always generated when the button is pressed.

NotifyHost Event, if is it enabled, will interrupt (notify) the host in 2 different modes:

- **OnChange**: for momentary button, when the button is released (OnRelease) and for latching button, when the button is pressed (OnPress).

- **OnPressAndHold**: when the button is first pressed (OnPress), as long as the button is being pressed (OnHold) and then when the button is released (OnRelease). This mode is very useful when a button is used to increase or decrease a value.

Properties

Name	Index	Type	Access	Description
Style	10	Byte	R	Button Style 0 = Bitmap 1 = BitmapAndLabel 2 = Clear 3 = Standard 4 = Simple
Type	11	Byte	R	Button Type 0 = Momentary 1 = Latching
Font	12	Byte	R/W	Label Font ID
Alignment	13	Byte	R	Label Alignment (Position) 0 = Center 1 = Other (LabelPosX and LabelPosY)
AlphaBlendingRatio	14	Byte	R	Alpha Blending Ratio 0 = 0% (No Alpha blending) 1 = 50%
BorderStyle	46	Byte	R	Border Style (Only for Simple Style) 0 = None 1 = Flat
State	43	Byte	R/W	Button State 0 = Not Pressed or Up 1 = Pressed or Down 2 = Disabled
Label	64	String	W	Button Label. Up to 64 ASCII characters including the termination code “NULL”
UpBitmapID	17	Word	R/W	Up or NotPressed Bitmap ID
DownBitmapID	19	Word	R/W	Down or Pressed Bitmap ID
DisableBitmapID	21	Word	R/W	Disable Bitmap ID
LableColor	23	Word	R/W	Label Color in RGB565 format
AlphaBlendingColor	25	Word	R	Alpha Blending Color in RGB565 format
BorderColor	47	Word	R/W	Border Color (Only for Simple Style) in RGB565 format
InactiveBackColor	49	Word	R/W	Inactive (Not Pressed) Back Color (Only for Simple Style) in RGB565 format
ActiveBackColor	51	Word	R/W	Active (Pressed) Back Color (Only for Simple Style) in RGB565 format
AcitveLabelColor	53	Word	R/W	Active (Pressed) Label Color (Only for Simple Style) in RGB565 format
LabelPosX	31	Byte	R	Label X Position relative to button Left
LabelPosY	32	Byte	R	Label Y Position relative to button Top
OffsetX	15	Byte	R	Width Offset
OffsetY	16	Byte	R	Height Offset
TouchZone	27	Byte	R	Enable/Disable Touch zone 0 = Disabled 1 = Enabled
TouchZoneOffset	28	Byte	R/W	Touch Zone Offset
SoundEvent	29	Byte	R/W	Enable/Disable Sound Event 0 = Disabled 1 = Enabled

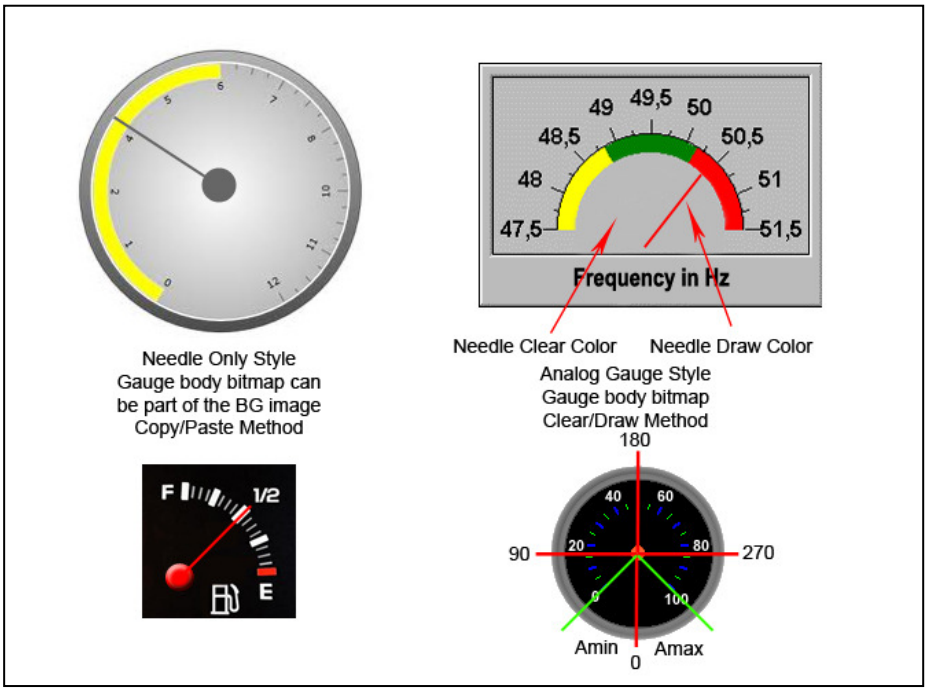
NotifyHostEvent	30	Byte	R/W	Set NotifyHost Event 0 = Disabled 1 = OnChange 2 = OnPressAndHold
ToggleStateEvent	31	Byte	R/W	Enable/Disable Toggle State Event 0 = Disabled 1 = Enabled (Automatically redraw the button)
BeepDuration	45	Byte	R/W	Sound Period in msec
FunCode	34	Byte	R/W	Function Code. Not used.
FunVal0	35	Byte	R/W	Function Value 0. Not used.
FunVal1	36	Byte	R/W	Function Value 1. Not used.
FunVal2	37	Byte	R/W	Function Value 2. Not used.
FunVal3	38	Byte	R/W	Function Value 3. Not used.
FunVal4	39	Byte	R/W	Function Value 4. Not used.
FunVal5	40	Byte	R/W	Function Value 5. Not used.
FunVal6	41	Byte	R/W	Function Value 6. Not used.
FunVal7	42	Byte	R/W	Function Value 7. Not used.

Button GUI Commands

See “GUI Commands” For more Information.

GUI Command	**Description**
Button_State	Set button state
Button_Label	Set button label of caption

20.4. Needle:



The Needle object is used to create radial gauge which visually displays where a value lies in a range. There are two styles to create a radial gauge:

- **Needle Only**: in this style, only the needle is used to create the gauge. The gauge body bitmap is a part of the background image. The method used to rotate the needle is Copy/Paste method. In this method, needle new position pixels are copied to internal buffer, needle is drawn using the needle color and previous needle position pixels are pasted back to the background image. Up to 8 needle objects can be placed in a form at a time. Only thin line type can be used for this style.
- **Analog Gauge**: in this style, the needle and body bitmap are used to create the gauge. Clear/Draw method is used to rotate the needle. This is done by drawing the needle at the new position using needle color and clear or erase the previous one using the clear color. There is no limit on how many needle objects a form can have and both types (thin and thick) can be used.
  The area of the body bitmap where the needle will be drawn must have one solid color and the needle cannot be extended over the gauge scale or labels.

The needle range is:
Needle Rang = 360° – (Amin + Amax)
Where:
Amin: is the minimum angle.
Amax: is the maximum angle.

Properties

Name	Index	Type	Access	Description
Style	10	Byte	R	Needle Style 0 = Only Needle 1 = Analog Gauge
Type	11	Byte	R	Needle Type 0 = Normal line 1 = Thick line
MinAngle	12	Byte	R	Minimum angle in deg.

MaxAngle	13	Byte	R	Maximum angle in deg.
Length	14	Byte	R	Needle Length
PosX	15	Byte	R	Needle X Position relative to body bitmap Left (Only for Analog Gauge Style)
PosY	16	Byte	R	Needle Y Position relative to body bitmap Top (Only for Analog Gauge Style)
DrawColor	18	Word	R	Needle Draw Color in RGB565 format
ClearColor	20	Word	R	Needle Clear Color in RGB565 format (Only for Analog Gauge Style)
BodyBitmapID	22	Word	R	Gauge Body Bitmap ID (Only for Analog Gauge Style)
Value	37	Word	R/W	Needle Value. From 0 to Range in deg.
TouchZone	24	Byte	R	Enable/Disable Touch zone 0 = Disabled 1 = Enabled
TouchZoneOffset	25	Byte	R/W	Touch Zone Offset
SoundEvent	26	Byte	R/W	Enable/Disable Sound Event 0 = Disabled 1 = Enabled
NotifyHostEvent	27	Byte	R/W	Set NotifyHost Event 0 = Disabled 1 = Enabled (OnPress)
BeepDuration	42	Byte	R/W	Sound Period in msec
FunCode	28	Byte	R/W	Function Code. Not used.
FunVal0	29	Byte	R/W	Function Value 0. Not used.
FunVal1	30	Byte	R/W	Function Value 1. Not used.
FunVal2	31	Byte	R/W	Function Value 2. Not used.
FunVal3	32	Byte	R/W	Function Value 3. Not used.
FunVal4	33	Byte	R/W	Function Value 4. Not used.
FunVal5	34	Byte	R/W	Function Value 5. Not used.
FunVal6	35	Byte	R/W	Function Value 6. Not used.
FunVal7	36	Byte	R/W	Function Value 7. Not used.

Needle GUI Commands

See “GUI Commands” For more Information.

GUI Command	**Description**
Needle_Value	Set Needle Value

20.5. NumberBox:

NumberBox object is used to display up to 8-digits numerical data. Digits and other symbols are made from user defined bitmaps saved to the flash memory. The NumberBox object can display the input data in five different ways:

- **Unsigned Long**: number (0 to 99999999) will be converted to BCD and then display corresponding bitmaps without the sign.
- **(+/-) Signed Long**: number (-9999999 to +9999999) will be converted to BCD and then display corresponding bitmaps with (+) sign for positive number and (-) for negative number.
- **(-) Signed Long**: number (-9999999 to +9999999) will be converted to BCD and then display corresponding bitmaps with (-) sign for negative number and blank for positive number.
- **Time (hh:mm AM/PM)**: 5-digit time data will be converted to BCD and then display corresponding bitmaps according to this format "HH:MM AM/PM". For example: a value of 12341 will show 12:34PM, a value of 12340 will show 12:34AM. The first digit is for AM/PM (0 = AM, 1 = PM). The second and third digits for the minutes (0 to 59). The fourth and fifth digits are for the hours (1 to 12).
  To display "01:59AM", time decimal number = 0 + 59*100 + 1*1000 = 1590.
  To display "01:59PM", time decimal number = 1 + 59*100 + 1*1000 = 1591.
- **Date (mm/dd)**: 4-digit date data will be converted to BCD and then display corresponding bitmaps according to this format "MM/DD". The first and second digits are for the month (1 to 12) while the third and fourth are for the day (1 to 31).
  To display "11/12", date decimal number = 12 + 11*100 + 12 = 1112

Each NumberBox object has a bitmap array that hold the bitmap IDs of the digits and other symbols. All bitmaps must have the same height but they can have different width. For example, the width of the decimal point bitmap can be a smaller than the other digit bitmaps.

Bitmap Array Index	Bitmap ID	Example
0	Digit 0 Bitmap ID	0
1	Digit 1 Bitmap ID	1
2	Digit 2 Bitmap ID	2
3	Digit 3 Bitmap ID	3
4	Digit 4 Bitmap ID	4

5	Digit 5 Bitmap ID	5
6	Digit 6 Bitmap ID	6
7	Digit 7 Bitmap ID	7
8	Digit 8 Bitmap ID	8
9	Digit 9 Bitmap ID	9
10	Blank Bitmap ID	
11	. : or / Bitmap ID	.
12	+ or AM Bitmap ID	+
13	- or PM Bitmap ID	-

Properties

Name	Index	Type	Access	Description
Style	10	Byte	R	NumberBox Style 0 = Bitmap
Type	11	Byte	R	NumberBox Input Type 0 = Unsigned Long 1 = (+/-) Signed Long 2 = (-) Signed Long 3 = Time (hh:mm AM/PM) 4 = Date (mm:dd)
DigitCount	12	Byte	R	NumberBox digit count 0 = 1 digit 1 = 2 digits 2 = 3 digit 3 = 4 digits 4 = 5 digit 5 = 6 digits 6 = 7 digit 7 = 8 digits
DecPoint	13	Byte	R/W	NumberBox Decimal Point Position 0 = No Decimal Point 1 = 0.0 2 = 0.00 3 = 0.000 4 = 0.0000 5 = 0.00000 6 = 0.000000 7 = 0.0000000
LeadingZeros	14	Byte	R/W	Remove Leading Zeros 0 = Do not remove leading zeros 1 = Remove leading zeros
OffsetX	15	Byte	R	Width Offset
OffsetY	16	Byte	R	Height Offset
BitmapArray[0]	17	Word	R/W	Digit 0 Bitmap ID
BitmapArray[1]	19	Word	R/W	Digit 0 Bitmap ID
BitmapArray[2]	21	Word	R/W	Digit 0 Bitmap ID
BitmapArray[3]	23	Word	R/W	Digit 0 Bitmap ID
BitmapArray[4]	25	Word	R/W	Digit 0 Bitmap ID
BitmapArray[5]	27	Word	R/W	Digit 0 Bitmap ID
BitmapArray[6]	29	Word	R/W	Digit 0 Bitmap ID
BitmapArray[7]	31	Word	R/W	Digit 0 Bitmap ID
BitmapArray[8]	33	Word	R/W	Digit 0 Bitmap ID
BitmapArray[9]	35	Word	R/W	Digit 0 Bitmap ID

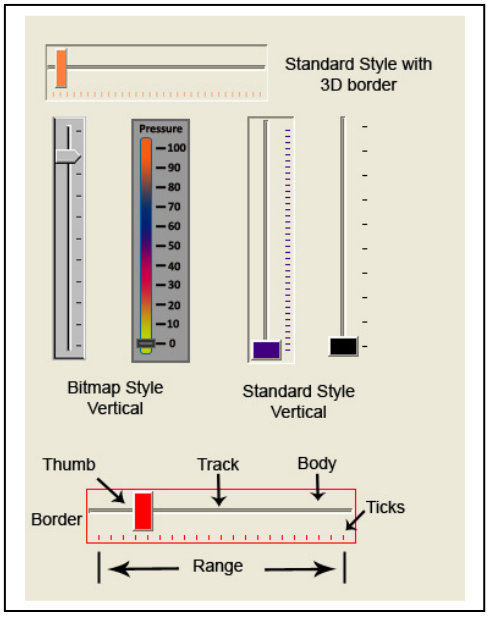
BitmapArray[10]	37	Word	R/W	Blank Bitmap ID
BitmapArray[11]	39	Word	R/W	. : or / Bitmap ID
BitmapArray[12]	41	Word	R/W	+ or AM Bitmap ID
BitmapArray[13]	43	Word	R/W	- or PM Bitmap ID
Value	58	DWord	R/W	NumberBox value.
TouchZone	45	Byte	R	Enable/Disable Touch zone 0 = Disabled 1 = Enabled
TouchZoneOffset	46	Byte	R/W	Touch Zone Offset
SoundEvent	47	Byte	R/W	Enable/Disable Sound Event 0 = Disabled 1 = Enabled
NotifyHostEvent	48	Byte	R/W	Set Notify Host Event 0 = Disabled 1 = Enabled (OnPress)
BeepDuration	62	Byte	R/W	Sound Period in msec
FunCode	49	Byte	R/W	Function Code. Not used.
FunVal0	50	Byte	R/W	Function Value 0. Not used.
FunVal1	51	Byte	R/W	Function Value 1. Not used.
FunVal2	52	Byte	R/W	Function Value 2. Not used.
FunVal3	53	Byte	R/W	Function Value 3. Not used.
FunVal4	54	Byte	R/W	Function Value 4. Not used.
FunVal5	55	Byte	R/W	Function Value 5. Not used.
FunVal6	56	Byte	R/W	Function Value 6. Not used.
FunVal7	57	Byte	R/W	Function Value 7. Not used.

NumberBox GUI Commands

See “GUI Commands” For more Information.

GUI Command	**Description**
NumberBox_Value	Set Numberbox Value

20.6. Slider:



Slider is a control that lets the user to select a value from a range of values by moving a Thumb along a track. Slider object has two different styles:

- **Bitmap**: slider is created from three bitmaps. One for the slider body, the second for the Inactive thumb and the third bitmap for the Active thumb (optional). Slider can be horizontal or vertical and there is no border or ticks for this style.
- **Standard**: this style is similar to Window's slider and does need any bitmaps. Slider can be horizontal or vertical and can have a border and/or ticks.

NotifyHost Event, if is it enabled, will interrupt (notify) the host in 2 different modes:

- **OnRelease**: when the slider thumb is released.
- **OnChange**: as long as the slider thumb is moving along the track.

Properties

Name	Index	Type	Access	Description
Style	10	Byte	R	Slider Style 0 = Bitmap 1 = Standard
Orientation	11	Byte	R	Slider Orientation 0 = Horizontal 1 = Vertical
Ticks	12	Byte	R	Slider Ticks 0 = No 1 = Yes
BorderStyle	13	Byte	R	Slider Border Style 0 = None 1 = Flat 2 = 3D
Range	14	Word	R	Slider Range
Change	16	Byte	R	Change. Not used.
ThumbWidth	18	Byte	R	Thumb Width (Only for Standard Style)

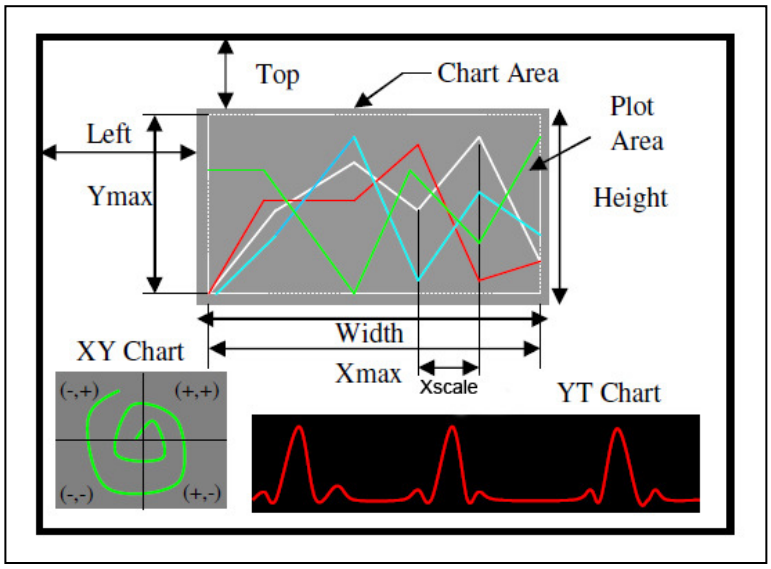
ThumbHeight	19	Byte	R	Thumb Height (Only for Standard Style)
ThumbPosX	20	Byte	R	Thumb Bitmap X position offset relative to Slider Left (Only for Bitmap Style)
ThumbPosY	21	Byte	R	Thumb Bitmap Y position offset relative to Slider Top (Only for Bitmap Style)
BodyBitmapID	22	Word	R/W	Slider Body Bitmap ID (Only for Bitmap Style)
InaThumbBitmap ID	24	Word	R/W	Inactive Thumb Bitmap ID (Only for Bitmap Style)
ActThumbBitmapID	26	Word	R/W	Active Thumb Bitmap ID (Only for Bitmap Style)
BackColor	28	Word	R/W	Slider Back Color. (Only for Standard Style)
ForeColor	30	Word	R/W	Slider Draw Color. (Only for Standard Style)
BorderColor	32	Word	R/W	Slider Border Color. (Only for Standard Style)
Value	48	Word	R/W	Slider Value from 0 to Slider Range - 1
TouchZone	34	Byte	R	Enable/Disable Touch zone 0 = Disabled 1 = Enabled
TouchZoneOffset	35	Byte	R/W	Touch Zone Offset
SoundEvent	36	Byte	R/W	Enable/Disable Sound Event 0 = Disabled 1 = Enabled
NotifyHostEvent	37	Byte	R/W	Set NotifyHost Event 0 = Disabled 1 = Enabled (OnPress)
MoveThumbEvent	38			Enable/Disable Thumb Move Event 0 = Disabled 1 = Enabled (Automatically draw the slider thumb at the new position)
BeepDuration	53	Byte	R/W	Sound Period in msec
FunCode	39	Byte	R/W	Function Code. Not used.
FunVal0	40	Byte	R/W	Function Value 0. Not used.
FunVal1	41	Byte	R/W	Function Value 1. Not used.
FunVal2	42	Byte	R/W	Function Value 2. Not used.
FunVal3	43	Byte	R/W	Function Value 3. Not used.
FunVal4	44	Byte	R/W	Function Value 4. Not used.
FunVal5	45	Byte	R/W	Function Value 5. Not used.
FunVal6	46	Byte	R/W	Function Value 6. Not used.
FunVal7	47	Byte	R/W	Function Value 7. Not used.

Slider GUI Commands

See “GUI Commands” For more Information.

GUI Command	**Description**
Slider_Value	Set Slider Value

20.7. Chart:



The Chart object allows displaying multiple traces or graphs at the same time. There are two types of the chart object:

- Y-Time (YT): it is similar to an oscilloscope YT mode. The Y-axis represents the value while the X-axis represents the time. A new line is drawn at every new point (Y) and scroll back to the chart origin (0, 0) when the chart is full.
- X-Y (XY): it is similar to am oscilloscope XY mode. The Y-axis represents the Y value while the X-axis represents the X value. The origin (0, 0) of the XY chart is at the center and the range for the X and Y values is:
  -Xmax/2 <= X <= Xmax/2
  -Ymax/2 <= Y <= Ymax/2

Up to 4 traces can be displayed at the same time and each trace can have its own color. All chart traces must be updated at the same time.

Properties

Name	Index	Type	Access	Description
Style	10	Byte	R	Chart Style 0 = Line
Type	11	Byte	R	Chart Type 0 = YT 1 = XY
TraceCount	12	Byte	R	Chart Trace Count 0 = 1 1 = 2 2 = 3 3 = 4
GridStyle	13	Byte	R	Chart Grid Style. Not used
PenWidth	14	Word	R	Draw Width. Fixed to thin (1 pixel) line width.
BorderStyle	15	Byte	R	Chart Border Style 0 = None 1 = Flat 2 = 3D
Xmax	16	Word	R	The maximum value of the X-axis

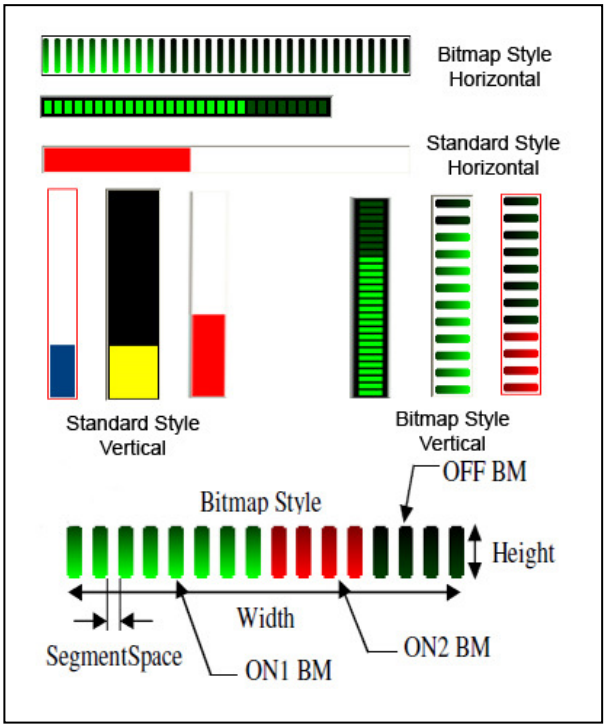
Ymax	18	Word	R	The maximum value of the Y-axis
Xscale	20	Byte	R	The scale of the X-axis. Default = 1
Yscale	21	Byte	R	The scale of the Y-axis. Fixed at 1.
Xgrid	22	Word	R/W	Not used.
Ygrid	23	Word	R/W	Not used
OffsetX	24	Word	R/W	Chart width offset.
OffsetY	25	Word	R/W	Chart height offset.
BackColor	26	Word	R/W	Chart Back Color in RGB565 format
BorderColor	28	Word	R/W	Chart Border Color in RGB565 format
GridColor	30	Word	R/W	Chart Grid Color in RGB565 format
Trace0Color	32	Word	R/W	Chart Trace0 Color in RGB565 format
Trace1Color	34	Word	R/W	Chart Trace1 Color in RGB565 format
Trace2Color	36	Word	R/W	Chart Trace2 Color in RGB565 format
Trace3Color	38	Word	R/W	Chart Trace3 Color in RGB565 format
TouchZone	40	Byte	R	Enable/Disable Touch zone 0 = Disabled 1 = Enabled
TouchZoneOffset	41	Byte	R/W	Touch Zone Offset
SoundEvent	42	Byte	R/W	Enable/Disable Sound Event 0 = Disabled 1 = Enabled
NotifyHostEvent	43	Byte	R/W	Set NotifyHost Event 0 = Disabled 1 = Enabled (OnPress)
BeepDuration	53	Byte	R/W	Sound Period in msec
FunCode	44	Byte	R/W	Function Code. Not used.
FunVal0	45	Byte	R/W	Function Value 0. Not used.
FunVal1	46	Byte	R/W	Function Value 1. Not used.
FunVal2	47	Byte	R/W	Function Value 2. Not used.
FunVal3	48	Byte	R/W	Function Value 3. Not used.
FunVal4	49	Byte	R/W	Function Value 4. Not used.
FunVal5	50	Byte	R/W	Function Value 5. Not used.
FunVal6	51	Byte	R/W	Function Value 6. Not used.
FunVal7	52	Byte	R/W	Function Value 7. Not used.

Chart GUI Commands

See “GUI Commands” For more Information.

GUI Command	**Description**
Chart_AddPoint	Add new point to a chart
Chart_Clear	Clear the chart
Chart_HCL	Draw a Horizontal Constant Line to a chart
Chart_VCL	Draw a Vertical Constant Line to a chart

20.8. BarGraph:



BarGraph or ProgessBar object is used to display the progress of an operation. The typical visual appearance is a bar that animates a filled area as progress continues.
There are two styles of the BarGraph object:

- **Bitmap**: the bar segments are made from bitmaps. One bitmap for OFF segment, the second for ON1 segment and the third for ON2 segment (optional). The "SegmentCount" and the "Segment space" properties set the BarGraph width (horizontal) or height (vertical). When the value of the bargraph is zero, then the OFF bitmaps will be displayed. If the value is higher than zero, then ON1 or ON2 bitmaps will be displayed based on "BitmapIndex" property. The space between the segments is transparent.
- **Standard**: this style is similar to Window's ProgressBar. The width or height of a bar will be used to indicate the operation progress or value. Only Bar type is available for this style.

The Bitmap Style BarGraph displays the progress in two different ways or types:

- **Bar**: the OFF segments will be replaced by ON1 or ON2 segments depending on the value. For example, if the value = 5, then the first 5 segments will be ON.
- **Dot**: Only one OFF segment will be replaced by ON1 or ON2 segment depending on the value. For example, if the value = 5, then the fifth segment only will be ON

The BarGraph can be horizontally oriented with progress starts from left to right or vertically oriented with progress starts from bottom to top.

Properties

Name	Index	Type	Access	Description
Style	10	Byte	R	BarGraph Style 0 = Bitmap 1 = Standard
Type	11	Byte	R	BarGraph Type

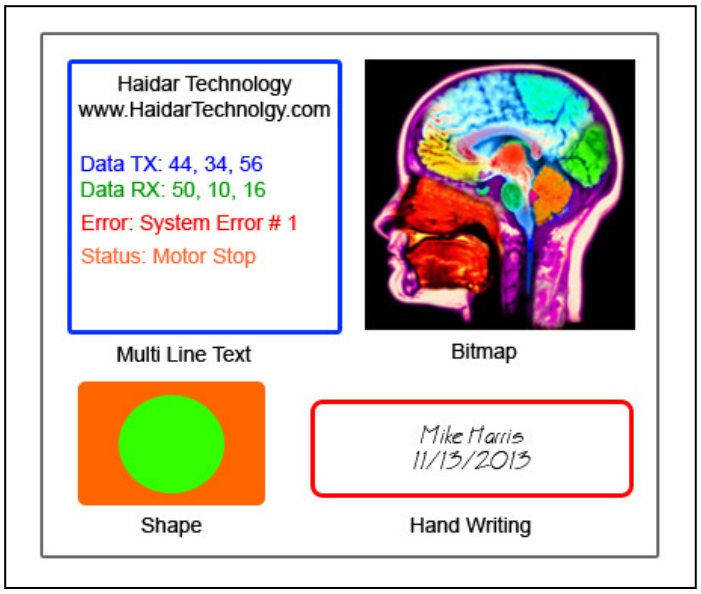
				0 = Bar 1 = Dot
Orientation	12	Byte	R	BarGraph Orientation 0 = Horizontal 1 = Vertical
BorderStyle	13	Byte	R	BarGraph Border Style 0 = None 1 = Flat 2 = 3D
BackStyle	14	Byte	R	Not Used.
SegCount	15	Word	R	BarGraph Segment Count
SegSpace	17	Byte	R	Distance between segments (Only for Bitmap Style)
BackColor	18	Word	R/W	BarGraph Back Color in RGB565 format
ForeColor	20	Word	R/W	BarGraph Draw Color in RGB565 format
BorderColor	22	Word	R/W	BarGraph Border Color in RGB565 format
OffsetX	24	Byte	R	Width Offset.
OffsetY	25	Byte	R	Height Offset.
OFFBitmapID	26	Word	R/W	"OFF" Segment Bitmap ID
ON1BitmapID	28	Word	R/W	"ON1" Segment Bitmap ID
ON2BitmapID	30	Word	R/W	"ON2" Segment Bitmap ID
Value	36	Word	R/W	BarGraph value from 0 to Segment Count
BitmapIndex	38	Byte	R/W	Segment Bitmap Index 0 = OFF Bitmap 1 = ON1 Bitmap 2 = ON2 Bitmap
TouchZone	32	Byte	R	Enable/Disable Touch zone 0 = Disabled 1 = Enabled
TouchZoneOffset	33	Byte	R/W	Touch Zone Offset
SoundEvent	34	Byte	R/W	Enable/Disable Sound Event 0 = Disabled 1 = Enabled
NotifyHostEvent	35	Byte	R/W	Set NotifyHost Event 0 = Disabled 1 = Enabled (OnPress)
BeepDuration	48	Byte	R/W	Sound Period in msec
FunCode	39	Byte	R/W	Function Code. Not used.
FunVal0	40	Byte	R/W	Function Value 0. Not used.
FunVal1	41	Byte	R/W	Function Value 1. Not used.
FunVal2	42	Byte	R/W	Function Value 2. Not used.
FunVal3	43	Byte	R/W	Function Value 3. Not used.
FunVal4	44	Byte	R/W	Function Value 4. Not used.
FunVal5	45	Byte	R/W	Function Value 5. Not used.
FunVal6	46	Byte	R/W	Function Value 6. Not used.
FunVal7	47	Byte	R/W	Function Value 7. Not used.

BarGraph GUI Commands

See "GUI Commands" For more Information.

GUI Command	**Description**
BarGraph_Value	Set BarGraph value

20.9. PictureBox:



Picture Box is basically an area of a form in which you can print text, draw shapes, display bitmaps, stylus hand writing and a lot more.
The back style of a picture box can be set to a bitmap (background image) or solid background color.
The clipping region width and height (drawing area) of the Picture Box is defined as:
W = PictureBox Width – 2*OffsetX
H = PictureBox Height – 2*OffsetY
Any graphics outside this region will be clipped.

Properties

Name	Index	Type	Access	Description
BackStyle	10	Byte	R/W	PictureBox Back Style 0 = Bitmap 1 = Solid back color
DrawType	11	Byte	R/W	Picture Box Draw Type 0 = Solid 1 = Dash 2 = Dot
DrawWidth	12	Byte	R/W	Picture Box Draw Width 0 = Thin (1 pixel) 1 = Thick (3 pixels)
Font	13	Byte	R/W	Picture Box Font from 0 to 7
BorderStyle	14	Byte	R	Picture Box Border Style 0 = None 1 = Flat 2 = 3D
BackColor	15	Word	R/W	Picture Box Back Color in RGB565 format
ForeColor	17	Word	R/W	Picture Box Draw Color in RGB565 format
BorderColor	19	Word	R/W	Picture Box Border Color in RGB565 format
OffsetX	21	Byte	R	Width Offset.
OffsetY	22	Byte	R	Height Offset.
BGBitmapID	23	Word	R/W	Picture Box Background bitmap
TouchZone	25	Byte	R	Enable/Disable Touch zone

				0 = Disabled 1 = Enabled
TouchZoneOffset	26	Byte	R/W	Touch Zone Offset
SoundEvent	27	Byte	R/W	Enable/Disable Sound Event 0 = Disabled 1 = Enabled
NotifyHostEvent	28	Byte	R/W	Set NotifyHost Event 0 = Disabled 1 = Enabled (OnPress)
SketchEvent	29			Enable/Disable Sketch (stylus drawing) Event 0 = Disabled 1 = Enabled
BeepDuration	39	Byte	R/W	Sound Period in msec
FunCode	30	Byte	R/W	Function Code. Not used.
FunVal0	31	Byte	R/W	Function Value 0. Not used.
FunVal1	32	Byte	R/W	Function Value 1. Not used.
FunVal2	33	Byte	R/W	Function Value 2. Not used.
FunVal3	34	Byte	R/W	Function Value 3. Not used.
FunVal4	35	Byte	R/W	Function Value 4. Not used.
FunVal5	36	Byte	R/W	Function Value 5. Not used.
FunVal6	37	Byte	R/W	Function Value 6. Not used.
FunVal7	38	Byte	R/W	Function Value 7. Not used.

PictureBox GUI Commands

See “GUI Commands” For more Information.

GUI Command	**Description**
PictureBox_Clear	Clear Picture Box
PictureBox_Bitmap	Display a bitmap from the flash memory
PictureBox_Line	Draw Line
PictureBox_Rectangle	Draw Rectangle
PictureBox_Circle	Draw Circle
PictureBox_SetPixel	Set a Pixel
PictureBox_GetPixel	Get a Pixel color
PictureBox_CopyArea	Copy an area of the Picture Box to the copy buffer
PictureBox_PasteArea	Paste to an area of the Picture Box from the copy buufer
PictureBox_Print	Print text
PictureBox_Picture	Display a bitmap directly from the serial port

20.10. Image:

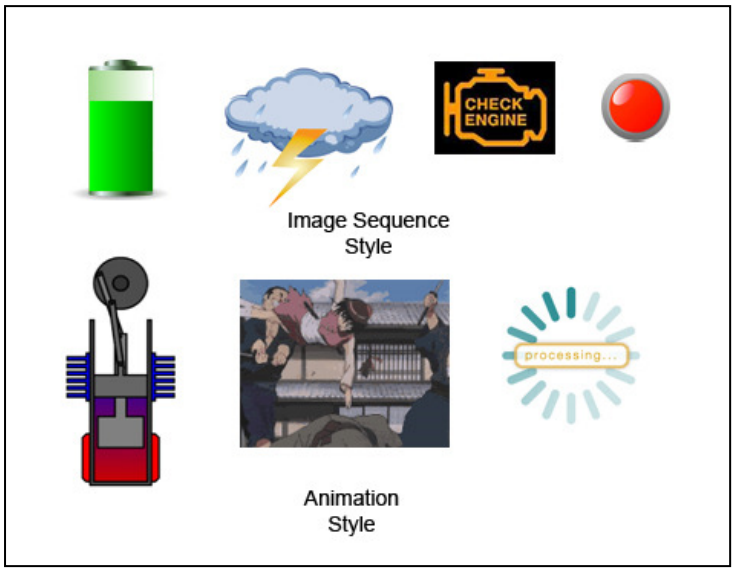


Image Object is a simple yet very powerful object. It is used to display one bitmap from an array (up to 48) of bitmaps.
There are two styles for the Image Object:

- **ImageSequence**: one bitmap form the object bitmap array will be displayed based on the object value property. For example, Value = 0, the first bitmap in the array will be displayed. This style can be used to show the progress or the status of an operation such as charging a battery, weather status, alarms… Below is an example of a charging battery indicator:

Value Property	Battery Power	Bitmap ID	Bitmap
0	90 to 100%	BitmapArray[0]	
1	80 to 89%	BitmapArray[1]	
2	70 to 79%	BitmapArray[2]	
3	60 to 69%	BitmapArray[3]	
4	50 to 59%	BitmapArray[4]	
5	40 to 49%	BitmapArray[5]	
6	30 to 39%	BitmapArray[6]	
7	20 to 29%	BitmapArray[7]	
8	0 to 19%	BitmapArray[8]	

- **Animation**: is the same as the Image Sequence style but the bitmaps (frames) in the list will be displayed rapidly to create the illusion of motion. The speed of the animation is controlled by an internal timer and the size of the frame. The smaller the frame is, the faster can be drawn. Animation is time consuming process and it will slow the processor down. Only one animation can be played at a time. Below is an example of rotating earth at 500msec or 2 frame/second:

Timer	Bitmap or Frame
0msec	
500msec	
1sec	
1.5sec	
2sec	
2.5sec	
3sec	
3.5sec	
4sec	
4.5sec	

Image object can also generate Touch Event when it is pressed by the user. If the Special Event is enabled, then the Image object will response to the user touch based on its style:

- For ImageSequence style, the object value will increment by 1 every time is pressed and the corresponding bitmap will be displayed. The value will roll back to zero when it reaches "BitmapCount".
- For Animation style, this will Play or Stop the animation.

Properties

Name	Index	Type	Access	Description
Style	10	Byte	R	Image Style 0 = ImageSequence 1 = Animation
Move	11	Byte	R	Not Used.
Speed	12	Byte	R/W	Animation Internal Timer Value 0 = 100msec 1 = 200msec 2 = 300msec 3 = 400msec 4 = 500msec 5 = 600msec 6 = 700msec 7 = 800msec 8 = 900msec 9 = 1000msec
BitmapCount	13	Byte	R/W	Number of bitmaps in the array. Maximum = 48
Value	14	Byte	R/W	Value
Bm[0] to Bm[47]	32 - 126	Word	R/W	Bitmap Array or List.
TouchZone	15	Byte	R	Enable/Disable Touch zone 0 = Disabled

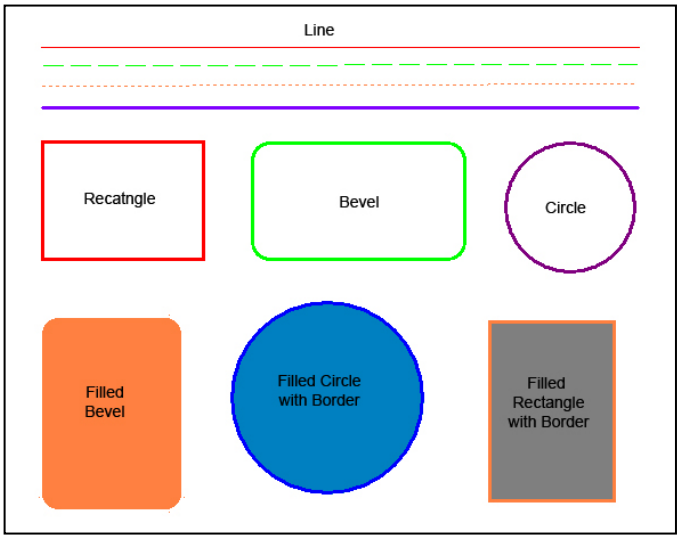
				1 = Enabled
TouchZoneOffset	16	Byte	R/W	Touch Zone Offset
SoundEvent	17	Byte	R/W	Enable/Disable Sound Event 0 = Disabled 1 = Enabled
NotifyHostEvent	18	Byte	R/W	Set NotifyHost Event 0 = Disabled 1 = Enabled (OnPress)
SpecialEvent	28	Byte	R/W	Set Special Event 0 = Disabled 1 = Increment/Play
BeepDuration	29	Byte	R/W	Sound Period in msec
FunCode	19	Byte	R/W	Function Code. Not used.
FunVal0	20	Byte	R/W	Function Value 0. Not used.
FunVal1	21	Byte	R/W	Function Value 1. Not used.
FunVal2	22	Byte	R/W	Function Value 2. Not used.
FunVal3	23	Byte	R/W	Function Value 3. Not used.
FunVal4	24	Byte	R/W	Function Value 4. Not used.
FunVal5	25	Byte	R/W	Function Value 5. Not used.
FunVal6	26	Byte	R/W	Function Value 6. Not used.
FunVal7	27	Byte	R/W	Function Value 7. Not used.

Image GUI Commands

See "GUI Commands" For more Information.

GUI Command	**Description**
Image_Value	Set Image Object Value
Animation_Play	Play Animation
Animation_Stop	Stop Animation

20.11. Shape:



The Shape Object is used to draw a shape element such as line, rectangle, circle … to a form.

Properties

Name	Index	Type	Access	Description
Style	10	Byte	R	Shape Style 0 = Line 1 = Rectangle 2 = Bevel 3 = Circle 4 = Filled Bevel 5 = Filled Circle 6 = Filled Rectangle
DrawType	11	Byte	R/W	Shape Draw Type 0 = Solid 1 = Dash 2 = Dot
DrawWidth	12	Byte	R/W	Shape Draw Width 0 = Thin 1 = Thick
BorderStyle	13	Byte	R/W	Shape Border Style 0 = None 1 = Flat
ForeColor	14	Word	R/W	Fore or Draw Color in RGB565 format
BorderColor	16	Word	R/W	Border Color in RGB565 format
OffsetX	18	Byte	R	Width Offset
OffsetY	19	Byte	R	Height Offset
TouchZone	20	Byte	R	Enable/Disable Touch zone 0 = Disabled 1 = Enabled
TouchZoneOffset	21	Byte	R/W	Touch Zone Offset
SoundEvent	22	Byte	R/W	Enable/Disable Sound Event 0 = Disabled 1 = Enabled
NotifyHostEvent	23	Byte	R/W	Set NotifyHost Event

				0 = Disabled 1 = Enabled (OnPress)
BeepDuration	33	Byte	R/W	Sound Period in msec
FunCode	24	Byte	R/W	Function Code. Not used.
FunVal0	25	Byte	R/W	Function Value 0. Not used.
FunVal1	26	Byte	R/W	Function Value 1. Not used.
FunVal2	27	Byte	R/W	Function Value 2. Not used.
FunVal3	28	Byte	R/W	Function Value 3. Not used.
FunVal4	29	Byte	R/W	Function Value 4. Not used.
FunVal5	30	Byte	R/W	Function Value 5. Not used.
FunVal6	31	Byte	R/W	Function Value 6. Not used.
FunVal7	32	Byte	R/W	Function Value 7. Not used.

21.GUI COMMANDS

GUI Commands are a set of serial commands that can be used by the host to interact with the objects. The table below lists all available GUI Commands.

Command Name	Code	Valid For	Command Description
Form_Load	100	Form	Load a Form
Form_Draw	101	Form	Draw a Form
Form_View	102	Form	View/Hide a Form
Form_Move	103	Form	Move Form to new position
Form_Enable	108	Form	Enable/Disable Form UI
Form_Status	111	Form	Get From UI status
Form_Redraw	114	Form	Redraw a Form
Get_ActiveScrID	104	All	Return Active Screen ID
Get_ActiveWin1ID	105	All	Return Active Window1 ID
Get_ActiveWin2ID	106	All	Return Active Window2 ID
Get_uiMessage	109	All	Return uiMessage
Get_uiMessageAdvance	117	All	Reaturn uiMessage and Touch X, Y
Get_uiStatus	110	All	Return UI Status
Enable_UI	107	All	Enable/Disable UI
Get_ObjProperty	112	All	Return Object Property Value
Set_ObjProperty	113	All	Set Object Property Value
Redraw_Object	115	All	Redraw Object
Chart_AddPoint	120	Chart	Add a new point to a Chart
Chart_Clear	121	Chart	Clear a Chart
Chart_HCL	122	Chart	Draw Horizontal Line to a Chart
Chart_VCL	123	Chart	Draw Vertical Line to a Chart
TextBox_Clear	124	TextBox	Clear TextBox
TextBox_State	125	TextBox	Set a TextBox State
TextBox_Text	126	TextBox	Display Text on a TextBox
TextBox_Number	127	TextBox	Display Number on a TextBox
BarGraph_Value	128	BarGraph	Set the Value of a BarGraph
Button_State	129	Button	Set the State of a Button
NumberBox_Value	130	NumberBox	Set the Value of a NumberBox
Needle_Value	131	Needle	Set the value of a Needle
Slider_Value	132	Slider	Set the Value of a Slider
Image_Value	162	Image	Set the Value of an Image
Image_Play	160	Image	Play Animation
Image_Stop	161	Image	Stop Animation
PictureBox_Clear	140	PictureBox	Clear a PictureBox
PictureBox_Bitmap	141	PictureBox	Display a Bitmap on a PictureBox
PictureBox_Line	142	PictureBox	Draw Line to a PictureBox
PictureBox_Rectangle	143	PictureBox	Draw Rectangle to a PictureBox
PictureBox_Circle	144	PictureBox	Draw Circle to a PictureBox
PictureBox_SetPixel	145	PictureBox	Draw Pixel to a PictureBox
PictureBox_GetPixel	146	PictureBox	Return the Color of a Pixel
PictureBox_CopyArea	147	PictureBox	Copy an Area of the PictureBox
PictureBox_PasteArea	148	PictureBox	Paste an Area to a PictureBox
PictureBox_Print	149	PictureBox	Write Text to a PictureBox
PictureBox_Picture	150	PictureBox	Display a Picture form the serial Port

Below is the description of each command in details. For clarity, comma (,) is used to separate between the command bytes or the response bytes.

21.1. Form_Load

Description	Loads form data into GUI buffer, but does not display it. You need to send "Form_Draw" after loading the form to display it. The host can modify the form object properties before displaying it.
Command Code	100
Command	DVID, 0, 2, 100, FormID, CS
Response	ACK/NAK
Arguments	FormID = Form ID number from 0 to 63.
Example	DVID, 0, 2, 100, 0, CS

21.2. Form_Draw

Description	Draws or shows the loaded form (active form) on the display.
Command Code	101
Command	DVID, 0, 2, 101 , FormID, CS
Response	ACK/NAK
Arguments	FormID = Form ID number from 0 to 63.
Example	DVID, 0, 2, 101, 0, CS

21.3. Form_View

Description	View or Hide a Form. Mostly used with Window1 and Window2 form type.
Command Code	102
Command	DVID, 0, 3, 102 , FormID, Value, CS
Response	ACK/NAK
Arguments	FormID = Form ID number from 0 to 63. Value = 0 => Hide Value = 1 => View
Example	DVID, 0, 3, 102, 1, CS

21.4. Form_Move

Description	Move form to new position (X, Y). Only valid for Window1 and Window2 form types. New form (X, Y) must be: X + Form Width ≤ LCDResX - 1 Y + From Height ≤ LCDResY - 1
Command Code	103
Command	DVID, 0, 6, 103, FormID, XH, XL, YH, YL, CS
Response	ACK/NAK
Arguments	FormID = Form ID number from 0 to 63. XH&XL: Form new X or Left position. YH&YL: Form new Y or Top position.
Example	Move Form#2 to (64, 64) DVID, 0, 6, 103, 2, 0, 64, 0, 64, CS

21.5. Get_ActiveScrID

Description	Returns the Active Screen (Loaded Screen) ID.
Command Code	104
Command	DVID, 0, 1, 104, CS
Response	ACK/NAK, 1, ID, CS
Arguments	ID : Active screen ID
Example	

21.6. Get_ActiveWin1ID

Description	Returns the Active Window1 (Loaded Window1) ID.
Command Code	105
Command	DVID, 0, 1, 105, CS
Response	ACK/NAK, 1, ID, CS
Arguments	ID : Active Window1 ID
Example	

21.7. Get_ActiveWin2ID

Description	Returns the Active Window2 (Loaded Window2) ID.
Command Code	106
Command	DVID, 0, 1, 106, CS
Response	ACK/NAK, 1, ID, CS
Arguments	ID : Active Window2 ID
Example	

21.8. Enable_UI

Description	Enable/Disable the User Interface of the whole GUI application. At power up or after reset, UI is enabled.
Command Code	107
Command	DVID, 0, 2, 107, Value, CS
Response	ACK/NAK
Arguments	Value = 0 => Disable UI Value = 1 => Enable UI
Example	

21.9. Form_Enable

Description	Enable/Disable the User Interface of on form.
Command Code	108
Command	DVID, 0, 3, 108, FormID, Value, CS
Response	ACK/NAK
Arguments	FormID, Form ID Number from 0 to 63. Value = 0 => Disable UI Value = 1 => Enable UI
Example	

21.10. Get_uiMessage

Description	Return the User Interface Message Data. Please see **User Interface Message**, page 38 for more information.
Command Code	109
Command	DVID, 0, 1, 109, CS
Response	ACK/NAK, 6, Status, Type, Code, ID, ValueH, ValueL, CS
Arguments	Status: Event Status. Type: Event Type. Code: Object Code. ID: Object ID. ValueH&ValueL: Object Value.
Example	

21.11. Get_uiMessageAdvance

Description	Return the User Interface Message Data and Touch Data. Please see **User Interface Message**, page 38 for more information.
Command Code	117
Command	DVID, 0, 1, 117, CS
Response	ACK/NAK, 6, Status, Type, Code, ID, ValueH, ValueL, Touh Status, Touch XH, Touch XL, Touch YH, Touch YL, 0, 0, CS
Arguments	Status: Event Status. Type: Event Type. Code: Object Code. ID: Object ID. ValueH&ValueL: Object Value. Touch Status = 0 => Pen Up = 1 => Pen Down Touch XH&XL: Touch X Coordinate Touch YH&YL: Touch Y Coordinate
Example	

21.12. Get_uiStatus

Description	Returns the User Interface status.
Command Code	110
Command	DVID, 0, 1, 110, CS
Response	ACK/NAK, 1, Status, CS
Arguments	Status = 0 => UI is Disabled Status = 1 => UI is Eanbled
Example	

21.13. Form_Status

Description	Returns the Form User Interface status.
Command Code	111
Command	DVID, 0, 2, 111, FromID, CS
Response	ACK/NAK, 1, Status, CS
Arguments	FormID: Form ID Number from 0 to 63 Status = 0 => UI is Disabled Status = 1 => UI is Eanbled
Example	

21.14. Get_ObjProperty

<table>
<tr><td>Description</td><td>Return the object property value as an array of 4 bytes [Byte0, Byte1, Byte2, Byte3].</td></tr>
<tr><td>Command Code</td><td>112</td></tr>
<tr><td>Command</td><td>DVID, 0, 4, 112, Code, ID, Index, CS</td></tr>
<tr><td>Response</td><td>ACK/NAK, 4, Byte0, Byte1, Byte2, Byte3, CS</td></tr>
<tr><td>Arguments</td><td>Code: Object code
ID: Object ID
Index: Object Property Index
Byte0: Property value byte 0 (MSB)
Byte1: Property value byte 1
Byte2: Property value byte 2
Byte3: Property value byte 3 (LSB)

<table>
<tr><td>Property Type</td><td>Byte0</td><td>Byte1</td><td>Byte 2</td><td>Byte 3</td></tr>
<tr><td>BYTE</td><td>V</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>WORD</td><td>VH</td><td>VL</td><td>0</td><td>0</td></tr>
<tr><td>DWORD</td><td>V3 MSB</td><td>V2</td><td>V1</td><td>V0 LSB</td></tr>
</table>
</td></tr>
<tr><td>Example</td><td>Byte Type Value:
Get the Font of a TextBox (Code = 1, ID = 20, Index = 12)
DVID,0 ,4, 104, 1, 20, 18, CS
Response: Font3
ACK, 4, 3, 0, 0, 0, 13

Word Type Value:
Get the BackColor of a TextBox (Code = 1, ID = 20, Index = 14)
DVID,0 ,4, 104, 1, 20, 21, CS
Response: White (RGB565 format)
ACK, 4, 255, 255, 0, 0, 8

DWord Type Value:
Get the Value of a NumberBox (Code = 4, ID = 8, Index = 58)
DVID,0 ,4, 104, 4, 8, 46, CS
Response: 99999999
ACK, 0x04, 0x05, 0xF5, 0xE0, 0xFF, 0xE3</td></tr>
</table>

21.15. Set_ObjProperty

<table>
<tr><td>Description</td><td>Set Object Property value as an array of 4 bytes [Byte0, Byte1, Byte2, Byte3] if the property type is Number (Byte, Word or DWord), or as an array of 64 characters if the property type is String.</td></tr>
<tr><td>Command Code</td><td>113</td></tr>
<tr><td>Command</td><td>Property type is Number (Property Index ≠ 64):
DVID, 0, 9, 113, Code, ID, Index, Mode, Byte0, Byte1, Byte2, Byte3, CS

Property type is a String (Property Index = 64):
DVID, 0, (5+n), 113, Code, ID, Index, Mode, CHR0, CHR1, …, CHRn, CS
Where n: is the string length (number of characters) including the termination character “NULL”. The maximum length is 64 characters (n = 63)</td></tr>
<tr><td>Response</td><td>ACK/NAK</td></tr>
<tr><td>Arguments</td><td>Code: Object code
ID: Object ID
Index: Object Property Index
Mode: Draw Mode
Mode = 0 => Do not redraw the object.</td></tr>
</table>

<table>
<tr><td></td><td>Mode = 1 => Redraw the object

Property Type is Number:
Byte0: Property value byte 0 (MSB)
Byte1: Property value byte 1
Byte2: Property value byte 2
Byte3: Property value byte 3 (LSB)

<table>
<tr><td>Property Type</td><td>Byte0</td><td>Byte1</td><td>Byte 2</td><td>Byte 3</td></tr>
<tr><td>BYTE</td><td>V</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>WORD</td><td>VH</td><td>VL</td><td>0</td><td>0</td></tr>
<tr><td>DWORD</td><td>V3 MSB</td><td>V2</td><td>V1</td><td>V0 LSB</td></tr>
</table>

Property Type is String:
CHR0 ….CHRn = The string ASCII codes. Maximum 64 and must be terminated by NULL (0x00). To start with new line, add 0x0A or “0/” to the end of the line.</td></tr>
<tr><td>Example</td><td>Set the Font of a TextBox to Font3 and redraw it after with the new font.
(Code = 1, ID = 20, Index = 12, Mode = 1)
DVID,0 ,9, 113, 1, 20, 12, 1, 3, 0, 0, 0, CS

Set the BackColor of a TextBox to Red (RGB565 format = 0xF800) and Redraw it with the new back color.
(Code = 1, ID = 20, Index = 14, Mode = 1)
DVID,0 ,9, 113, 1, 20, 14, 1, 248, 0, 0, 0, CS

Write “Hello” to a TextBox and Redraw it.
(Code = 1, ID = 20, Index = 64, Mode = 1)
DVID, 0, 11, 113, 1, 20, 64, 1, 72, 101, 108, 108, 111, 0, CS

Set the value of a NumberBox to -9999 and redraw it.
(Code = 4, ID = 8, Index = 58, Mode = 1)
DVID, 0, 9, 113, 4, 8, 58, 1, 255, 255, 216, 241, CS</td></tr>
</table>

21.16. From_Redraw

Description	Redraw a Form. This command will only redraw the objects that one or more of their properties have been modified by “Set_ObjPropery” command and the Draw Mode has been set to “0” or Do not redraw. This is very useful when you need to change the properties of a group of objects but you need to draw all of them at the same time. For example, you need to change the back color of a group of TextBoxes and then you need to redraw all of them with the new colors at the same time.
Command Code	114
Command	DVID, 0, 2, 114, FormID, CS
Response	ACK/NAK
Arguments	FormID: Form ID Number from 0 to 63.
Example	

21.17. Redraw_Object

Description	Redraw an Object.
Command Code	115
Command	DVID, 0, 3, 115, Code, ID, CS
Response	ACK/NAK
Arguments	Code: Object Code ID: Object ID

Example	

21.18. Chart_AddPoint

Description	Add a new point to a chart.
Command Code	120
Command	DVID, 1, 2, 120, ID, XS0H, XS0L, YS0H, YS0L, XS1H, XS1L, YS1H, YS1L, XS2H, XS2L, YS2H, YS2L, XS3H, XS3L, YS3H, YS3L, CS
Response	ACK/NAK
Arguments	ID: Chart ID Number. XS0,YS0: Trace0 (X, Y) Point XS1,YS1: Trace1 (X, Y) Point XS2,YS2: Trace2 (X, Y) Point XS3,YS3: Trace3 (X, Y) Point XH&XL: Point X coordinate (set to 0 if the chart type is YT) YH&YL: Point Y coordinate
Example	

21.19. Chart_Clear

Description	Clear Chart
Command Code	121
Command	DVID, 0, 2, 121, ID, CS
Response	ACK/NAK
Arguments	ID: Chart ID Number.
Example	

21.20. Chart_VCL

Description	Draw a constant vertical line to a chart
Command Code	123
Command	DVID, 0, 6, 123, ID, XH, XL, ColorH, ColorL, CS
Response	ACK/NAK
Arguments	ID: Chart ID Number XH&XL: Line X coordinate ColorH&ColorL: Line color in RGB565 format
Example	

21.21. Chart_HCL

Description	Draw a constant horizontal line to a chart
Command Code	122
Command	DVID, 0, 6, 122, ID, YH, YL, ColorH, ColorL, CS
Response	ACK/NAK
Arguments	ID: Chart ID Number XH&XL: Line Y coordinate ColorH&ColorL: Line color in RGB565 format
Example	

21.22. TextBox_Clear

Description	Clear a Text Box
Command Code	124
Command	DVID, 0, 2, 124, ID, CS
Response	ACK/NAK
Arguments	ID: Text Box ID Number
Example	

21.23. TextBox_State

Description	Set the Text Box State
Command Code	125
Command	DVID, 0, 2, 125, ID, State, CS
Response	ACK/NAK
Arguments	ID: Text Box ID Number State = 0 => Normal State = 1 => Highlight State = 2 => Invert
Example	

21.24. TextBox_Text

Description	Write text (up to 63 characters) to a Text Box.
Command Code	126
Command	DVID, 0, (2 + n), 126, ID, CHR0, CHR1 … CHRn, CS
Response	ACK/NAK
Arguments	ID: Text Box ID Number CHR0 ….CHRn: The string ASCII codes. Maximum 64 and must be terminated by NULL (0x00). To start with new line, add 0x0A or “0/” to the end of the line.
Example	

21.25. TextBox_Number

Description	Display a number on a Text Box. Number range from –99999999 to +99999999
Command Code	127
Command	DVID, 0, 8, 127, ID, LZ, DP, NumByte0, NumByte1, NumByte2, NumByte3, CS
Response	ACK/NAK
Arguments	ID: Text Box ID Number LZ : Leading Zeros 0 = Do not delete leading zeros 1 = Delete leading zeros DP: Decimal Point 0 = No Decimal Point 1 = 0.0 2 = 0.00 3 = 0.000 4 = 0.0000 5 = 0.00000 6 = 0.000000 7 = 0.0000000 NumByte0 = Number Byte0 (MSB) NumByte1 = Number Byte1 NumByte2 = Number Byte2

	NumByte3 = Number Byte3 (LSB)
Example	

21.26. BarGraph_Value

Description	Set the Value of a BarGraph
Command Code	128
Command	DVID, 0, 5, 128, ID, ValueH, ValueL, Index, CS
Response	ACK/NAK
Arguments	ID: BarGraph ID Number ValueH&ValueL: BarGraph Value Index: BarGraph Bitmap Index Index = 0 => OFF Bitmap Index = 1 => ON1 Bitmap Index = 2 => ON2 Bitmap
Example	

21.27. Button_State

Description	Set the State of a Button
Command Code	129
Command	DVID, 0, 2, 129, ID, State, CS
Response	ACK/NAK
Arguments	ID: Button ID Number State = 0 => Not Pressed State = 1 => Pressed State = 2 => Disable
Example	

21.28. Button_Label

Description	Update Button label of caption. Only valid for styles Bitmap and Label, Standard and Simple.
Command Code	163
Command	DVID, 0, (2 + n), 163, ID, CHR0, CHR1 … CHRn, CS
Response	ACK/NAK
Arguments	ID: Button ID Number CHR0 ….CHRn: The string ASCII codes. Maximum 64 and must be terminated by NULL (0x00). To start with new line, add 0x0A or “0/” to the end of the line.
Example	

21.29. NumberBox_Value

Description	Set the Value of a Number Box
Command Code	130
Command	DVID, 0, 6, 130, ID, Byte0, Byte1, Byte2, Byte3, CS
Response	ACK/NAK
Arguments	ID: NumberBox ID Number Byte0: Value Byte0 (MSB) Byte1: Value Byte1 Byte2: Value Byte2 Byte3: Value Byte3 (LSB)
Example	

21.30. Needle_Value

Description	Set the Value of a Needle
Command Code	131
Command	DVID, 0, 5, 131, ID, ValueH, ValueL, CS
Response	ACK/NAK
Arguments	ID: Needle ID Number ValueH&ValueL: Needle Value
Example	

21.31. Slider_Value

Description	Set the Value of a Slider
Command Code	131
Command	DVID, 0, 5, 131, ID, ValueH, ValueL, CS
Response	ACK/NAK
Arguments	ID: Slider ID Number ValueH&ValueL: Needle Value
Example	

21.32. Image_Value

Description	Set the Value of an Image (ImageSeq type only)
Command Code	162
Command	DVID, 0, 3, 162, ID, Value, CS
Response	ACK/NAK
Arguments	ID: Image ID Number Value: Image Value
Example	

21.33. Image_Play

Description	Play an Animation.
Command Code	160
Command	DVID, 0, 2, 160, ID, CS
Response	ACK/NAK
Arguments	ID: Image ID Number
Example	

21.34. Image_Stop

Description	Stop an Animation.
Command Code	161
Command	DVID, 0, 2, 161, ID, CS
Response	ACK/NAK
Arguments	ID: Image ID Number
Example	

21.35. PictureBox_Clear

Description	Clear a Picture Box
Command Code	140
Command	DVID, 0, 2, 140, ID, CS
Response	ACK/NAK
Arguments	ID: PictureBox ID Number
Example	

21.36. PictureBox_Bitmap

Description	Display a bitmap already saved to the flash memory.
Command Code	141
Command	DVID, 0, 10, 141, ID, BmIDH, BmIDL, R, M, XH, XL, YH, YL, CS
Response	ACK/NAK
Arguments	ID: PictureBox ID Number XH&XL: The X coordinate of the point at which the bitmap will be placed YH&YL: The Y coordinate of the point at which the bitmap will be placed BmIDH&BmIDL: Bitmap ID Number R: Bitmap Rotation or Orientation R = 0 => 0º R = 1 => 90º R = 2 => 180º R = 3 => 270º M: Bitmap Mirror M = 0 => Normal M = 1 => Mirror
Example	

21.37. PictureBox_Line

Description	Draw a Line to a Picture Box
Command Code	142
Command	DVID, 0, 10, 121, ID, X1H, X1L, Y1H, Y1L, X2H, X2L, Y2H, Y2L, CS
Response	ACK/NAK
Arguments	ID: PictureBox ID Number X1H&X1L: X1 coordinate Y1H&Y1L: Y1 coordinate X2H&X2L: X2 coordinate Y2H&Y2L: Y2 coordinate
Example	

21.38. PictureBox_Rectangle

Description	Draw a Rectangle to a Picture Box
Command Code	143
Command	DVID, 0, 12, 143, ID, X1H, X1L, Y1H, Y1L, X2H, X2L, Y2H, Y2L, R, F, CS
Response	ACK/NAK
Arguments	ID: PictureBox ID Number X1H&X1L: X1 coordinate Y1H&Y1L: Y1 coordinate X2H&X2L: X2 coordinate Y2H&Y2L: Y2 coordinate R: Round Corners

	0 = Straight corners 1 = Round corners F: Filled 0 = No Fill 1 = Filled
Example	

21.39. PictureBox_Circle

Description	Draw a Circle to a Picture Box
Command Code	144
Command	DVID, 0, 9, 124, ID, X0H, X0L, Y0H, Y0L, RadH, RadL, F, CS
Response	ACK/NAK
Arguments	ID: PictureBox ID Number X0H&X0L: X0 coordinate Y0H&Y0L: Y0 coordinate RadH&RadL: Radius F: Filled 0 = No Fill 1 = Filled
Example	

21.40. PictureBox_SetPixel

Description	Set the color a Pixel On a Picture Box
Command Code	145
Command	DVID, 0, 6, 129, ID, XH, XL, YH, YL, CS
Response	ACK/NAK
Arguments	ID: PictureBox ID Number XH&XL: Pixel X coordinate YH&YL: Pixel Y coordinate
Example	

21.41. PictureBox_GetPixel

Description	Return the color of a Picture Box Pixel
Command Code	146
Command	DVID, 0, 6, 146, ID, XH, XL, YH, YL, CS
Response	ACK/NAK, 2, ColorH, ColorL, CS
Arguments	ID: PictureBox ID Number XH&XL: Pixel X coordinate YH&YL: Pixel Y coordinate ColorH&ColorL: Pixel color in RGB565 format
Example	

21.42. PictureBox_CopyArea

Description	Copy an area (WXH) from a Picture Box at (X,Y) to the Copy Buffer
Command Code	147
Command	DVID, 0, 8, 147, ID, XH, XL, YH, YL, W, H, CS
Response	ACK/NAK
Arguments	ID: PictureBox ID Number XH&XL: X coordinate of the point at which the copy will start YH&YL: Y coordinate of the point at which the copy will start

	W: Area width H: Area height Note: W * H <= 1024
Example	

21.43. PictureBox_PasteArea

Description	Paste an area (WXH) from the Copy Buffer to a Picture Box at (X,Y)
Command Code	148
Command	DVID, 0, 8, 148, ID, XH, XL, YH, YL, W, H, CS
Response	ACK/NAK
Arguments	ID: PictureBox ID Number XH&XL: X coordinate of the point at which the paste will start YH&YL: Y coordinate of the point at which the paste will start W: Area width H: Area height Note: W * H <= 1024
Example	

21.44. PictureBox_Print

Description	Print a text (up to 255 characters) to a Picture Box starting from (X,Y)
Command Code	149
Command	DVID, (6 + n), 149, ID, XH, XL, YH, YL, CHR0, CHR1 … CHRn, CS Where n: is the text length (number of characters) including the termination character "NULL". The maximum length is 256 characters (n = 255)
Response	ACK/NAK
Arguments	ID: PictureBox ID Number XH&XL: X coordinate of the point at which the print will start YH&YL: Y coordinate of the point at which the print will start CHR0 …. CHRn: The text ASCII characters. The text must be terminated by NULL Character. To start with new line, add 0x0A or "0/" to the end of the line.
Example	

21.45. PictureBox_Picture

Description	Display a Picture (Bitmap) from the serial port to a Picture Box at (X,Y)
Command Code	150
Command	DVID, 1, 12, 150, ID, XH, XL, YH, YL, WidthH, WidthL, HeightH, HeightL, BlockH, BlockL, [Block Bytes up to 256 bytes], CS
Response	ACK/NAK
Arguments	ID: PictureBox ID Number XH&XL: The X coordinate of the point at which the bitmap will be placed YH&YL: The Y coordinate of the point at which the bitmap will be placed WidthH&WidthL: Picture Width HeightH&HeightL: Picture Height BlockH&BlockL: Data block number. Block Bytes: 256 bytes
Example	Display a bitmap (W = 16, H = 16) from the serial port on picture box (ID = 5) at (64,64) Total number of blocks = ((2 * W * H) /256)) = 2 => this command must be sent 2 times to fully display the bitmap. DVID, 1 ,12, 150, 5, 0, 64, 0, 64, 0, 16, 0, 16, 0, 0, [First block 256 bytes], CS DVID, 1 ,12, 150, 5, 0, 64, 0, 64, 0, 16, 0, 16, 0, 1, [Second block 256 bytes], CS

ASCII Character Codes:

- **ASCII control characters (character code 0-31)**

The first 32 characters in the ASCII-table are unprintable control codes and are used to control peripherals such as printers.

DEC	HEX	Symbol	Description
0	00	NUL	Null char
1	01	SOH	Start of Heading
2	02	STX	Start of Text
3	03	ETX	End of Text
4	04	EOT	End of Transmission
5	05	ENQ	Inquiry
6	06	ACK	Acknowledgment
7	07	BEL	Bell
8	08	BS	Back Space
9	09	HT	Horizontal Tab
10	0A	LF	Line Feed
11	0B	VT	Vertical Tab
12	0C	FF	Form Feed
13	0D	CR	Carriage Return
14	0E	SO	Shift Out / X-On
15	0F	SI	Shift In / X-Off
16	10	DLE	Data Line Escape
17	11	DC1	Device Control 1 (oft. XON)
18	12	DC2	Device Control 2
19	13	DC3	Device Control 3 (oft. XOFF)
20	14	DC4	Device Control 4
21	15	NAK	Negative Acknowledgement
22	16	SYN	Synchronous Idle
23	17	ETB	End of Transmit Block
24	18	CAN	Cancel
25	19	EM	End of Medium
26	1A	SUB	Substitute
27	1B	ESC	Escape
28	1C	FS	File Separator
29	1D	GS	Group Separator
30	1E	RS	Record Separator
31	1F	US	Unit Separator

- **ASCII printable characters (character code 32-127)**

Codes 32-127 are common for all the different variations of the ASCII table, they are called printable characters, represent letters, digits, punctuation marks, and a few miscellaneous symbols. You will find almost every character on your keyboard. Character 127 represents the command DEL.

DEC	HEX	Symbol	Description
32	20		Space
33	21	!	Exclamation mark
34	22	"	Double quotes (or speech marks)
35	23	#	Number
36	24	$	Dollar
37	25	%	Percent
38	26	&	Ampersand
39	27	'	Single quote
40	28	(	Open parenthesis (or open bracket)
41	29	)	Close parenthesis (or close bracket)
42	2A	*	Asterisk
43	2B	+	Plus
44	2C	,	Comma
45	2D	-	Hyphen
46	2E	.	Period, dot or full stop
47	2F	/	Slash or divide
48	30	0	Zero
49	31	1	One
50	32	2	Two
51	33	3	Three
52	34	4	Four
53	35	5	Five
54	36	6	Six
55	37	7	Seven
56	38	8	Eight
57	39	9	Nine
58	3A	:	Colon
59	3B	;	Semicolon
60	3C	<	Less than (or open angled bracket)
61	3D	=	Equals
62	3E	>	Greater than (or close angled bracket)
63	3F	?	Question mark
64	40	@	At symbol
65	41	A	Uppercase A
66	42	B	Uppercase B
67	43	C	Uppercase C
68	44	D	Uppercase D
69	45	E	Uppercase E
70	46	F	Uppercase F
71	47	G	Uppercase G
72	48	H	Uppercase H
73	49	I	Uppercase I
74	4A	J	Uppercase J
75	4B	K	Uppercase K
76	4C	L	Uppercase L
77	4D	M	Uppercase M

78	4E	N	Uppercase N
79	4F	O	Uppercase O
80	50	P	Uppercase P
81	51	Q	Uppercase Q
82	52	R	Uppercase R
83	53	S	Uppercase S
84	54	T	Uppercase T
85	55	U	Uppercase U
86	56	V	Uppercase V
87	57	W	Uppercase W
88	58	X	Uppercase X
89	59	Y	Uppercase Y
90	5A	Z	Uppercase Z
91	5B	[	Opening bracket
92	5C	\	Backslash
93	5D	]	Closing bracket
94	5E	^	Caret - circumflex
95	5F	_	Underscore
96	60	`	Grave accent
97	61	a	Lowercase a
98	62	b	Lowercase b
99	63	c	Lowercase c
100	64	d	Lowercase d
101	65	e	Lowercase e
102	66	f	Lowercase f
103	67	g	Lowercase g
104	68	h	Lowercase h
105	69	i	Lowercase i
106	6A	j	Lowercase j
107	6B	k	Lowercase k
108	6C	l	Lowercase l
109	6D	m	Lowercase m
110	6E	n	Lowercase n
111	6F	o	Lowercase o
112	70	p	Lowercase p
113	71	q	Lowercase q
114	72	r	Lowercase r
115	73	s	Lowercase s
116	74	t	Lowercase t
117	75	u	Lowercase u
118	76	v	Lowercase v
119	77	w	Lowercase w
120	78	x	Lowercase x
121	79	y	Lowercase y
122	7A	z	Lowercase z
123	7B	{	Opening brace
124	7C	|	Vertical bar
125	7D	}	Closing brace
126	7E	~	Equivalency sign - tilde
127	7F		Delete

- **The extended ASCII codes (character code 128-255)**

There are several different variations of the 8-bit ASCII table. The table below is according to ISO 8859-1, also called ISO Latin-1. Codes 129-159 contain the Microsoft® Windows Latin-1 extended characters.

DEC	HEX	Symbol	Description
128	80	€	Euro sign
129	81		
130	82	‚	Single low-9 quotation mark
131	83	*f*	Latin small letter f with hook
132	84	„	Double low-9 quotation mark
133	85	…	Horizontal ellipsis
134	86	†	Dagger
135	87	‡	Double dagger
136	88	ˆ	Modifier letter circumflex accent
137	89	‰	Per mille sign
138	8A	Š	Latin capital letter S with caron
139	8B	‹	Single left-pointing angle quotation
140	8C	Œ	Latin capital ligature OE
141	8D		
142	8E	Ž	Latin capital letter Z with caron
143	8F		
144	90		
145	91	‘	Left single quotation mark
146	92	’	Right single quotation mark
147	93	“	Left double quotation mark
148	94	”	Right double quotation mark
149	95	•	Bullet
150	96	–	En dash
151	97	—	Em dash
152	98	˜	Small tilde
153	99	™	Trade mark sign
154	9A	š	Latin small letter S with caron
155	9B	›	Single right-pointing angle quotation mark
156	9C	œ	Latin small ligature oe
157	9D		
158	9E	ž	Latin small letter z with caron
159	9F	Ÿ	Latin capital letter Y with diaeresis
160	A0		Non-breaking space
161	A1	¡	Inverted exclamation mark
162	A2	¢	Cent sign
163	A3	£	Pound sign
164	A4	¤	Currency sign
165	A5	¥	Yen sign
166	A6	¦	Pipe, Broken vertical bar
167	A7	§	Section sign
168	A8	¨	Spacing diaeresis - umlaut
169	A9	©	Copyright sign
170	AA	ª	Feminine ordinal indicator
171	AB	«	Left double angle quotes
172	AC	¬	Not sign
173	AD	-	Soft hyphen
174	AE	®	Registered trade mark sign

175	AF	¯	Spacing macron - overline
176	B0	°	Degree sign
177	B1	±	Plus-or-minus sign
178	B2	²	Superscript two - squared
179	B3	³	Superscript three - cubed
180	B4	´	Acute accent - spacing acute
181	B5	µ	Micro sign
182	B6	¶	Pilcrow sign - paragraph sign
183	B7	·	Middle dot - Georgian comma
184	B8	¸	Spacing cedilla
185	B9	¹	Superscript one
186	BA	º	Masculine ordinal indicator
187	BB	»	Right double angle quotes
188	BC	¼	Fraction one quarter
189	BD	½	Fraction one half
190	BE	¾	Fraction three quarters
191	BF	¿	Inverted question mark
192	C0	À	Latin capital letter A with grave
193	C1	Á	Latin capital letter A with acute
194	C2	Â	Latin capital letter A with circumflex
195	C3	Ã	Latin capital letter A with tilde
196	C4	Ä	Latin capital letter A with diaeresis
197	C5	Å	Latin capital letter A with ring above
198	C6	Æ	Latin capital letter AE
199	C7	Ç	Latin capital letter C with cedilla
200	C8	È	Latin capital letter E with grave
201	C9	É	Latin capital letter E with acute
202	CA	Ê	Latin capital letter E with circumflex
203	CB	Ë	Latin capital letter E with diaeresis
204	CC	Ì	Latin capital letter I with grave
205	CD	Í	Latin capital letter I with acute
206	CE	Î	Latin capital letter I with circumflex
207	CF	Ï	Latin capital letter I with diaeresis
208	D0	Ð	Latin capital letter ETH
209	D1	Ñ	Latin capital letter N with tilde
210	D2	Ò	Latin capital letter O with grave
211	D3	Ó	Latin capital letter O with acute
212	D4	Ô	Latin capital letter O with circumflex
213	D5	Õ	Latin capital letter O with tilde
214	D6	Ö	Latin capital letter O with diaeresis
215	D7	×	Multiplication sign
216	D8	Ø	Latin capital letter O with slash
217	D9	Ù	Latin capital letter U with grave
218	DA	Ú	Latin capital letter U with acute
219	DB	Û	Latin capital letter U with circumflex
220	DC	Ü	Latin capital letter U with diaeresis
221	DD	Ý	Latin capital letter Y with acute
222	DE	Þ	Latin capital letter THORN
223	DF	ß	Latin small letter sharp s - ess-zed
224	E0	à	Latin small letter a with grave
225	E1	á	Latin small letter a with acute
226	E2	â	Latin small letter a with circumflex
227	E3	ã	Latin small letter a with tilde

228	E4	ä	Latin small letter a with diaeresis
229	E5	å	Latin small letter a with ring above
230	E6	æ	Latin small letter ae
231	E7	ç	Latin small letter c with cedilla
232	E8	è	Latin small letter e with grave
233	E9	é	Latin small letter e with acute
234	EA	ê	Latin small letter e with circumflex
235	EB	ë	Latin small letter e with diaeresis
236	EC	ì	Latin small letter i with grave
237	ED	í	Latin small letter i with acute
238	EE	î	Latin small letter i with circumflex
239	EF	ï	Latin small letter i with diaeresis
240	F0	ð	Latin small letter eth
241	F1	ñ	Latin small letter n with tilde
242	F2	ò	Latin small letter o with grave
243	F3	ó	Latin small letter o with acute
244	F4	ô	Latin small letter o with circumflex
245	F5	õ	Latin small letter o with tilde
246	F6	ö	Latin small letter o with diaeresis
247	F7	÷	Division sign
248	F8	ø	Latin small letter o with slash
249	F9	ù	Latin small letter u with grave
250	FA	ú	Latin small letter u with acute
251	FB	û	Latin small letter u with circumflex
252	FC	ü	Latin small letter u with diaeresis
253	FD	ý	Latin small letter y with acute
254	FE	þ	Latin small letter thorn
255	FF	ÿ	Latin small letter y with diaeresis

Basic Colors:

Color	HTML/CSS Name	Hex Code #RRGGBB	Decimal Code (R,G,B)
	Black	#000000	(0,0,0)
	White	#FFFFFF	(255,255,255)
	Red	#FF0000	(255,0,0)
	Lime	#00FF00	(0,255,0)
	Blue	#0000FF	(0,0,255)
	Yellow	#FFFF00	(255,255,0)
	Cyan / Aqua	#00FFFF	(0,255,255)
	Magenta / Fuchsia	#FF00FF	(255,0,255)
	Silver	#C0C0C0	(192,192,192)
	Gray	#808080	(128,128,128)

	Maroon	#800000	(128,0,0)
	Olive	#808000	(128,128,0)
	Green	#008000	(0,128,0)
	Purple	#800080	(128,0,128)
	Teal	#008080	(0,128,128)
	Navy	#000080	(0,0,128)

Color	Color Name	Hex Code #RRGGBB	Decimal Code R,G,B
	maroon	#800000	(128,0,0)
	dark red	#8B0000	(139,0,0)
	brown	#A52A2A	(165,42,42)
	firebrick	#B22222	(178,34,34)
	crimson	#DC143C	(220,20,60)
	red	#FF0000	(255,0,0)
	tomato	#FF6347	(255,99,71)
	coral	#FF7F50	(255,127,80)
	indian red	#CD5C5C	(205,92,92)
	light coral	#F08080	(240,128,128)
	dark salmon	#E9967A	(233,150,122)
	salmon	#FA8072	(250,128,114)
	light salmon	#FFA07A	(255,160,122)
	orange red	#FF4500	(255,69,0)
	dark orange	#FF8C00	(255,140,0)
	orange	#FFA500	(255,165,0)
	gold	#FFD700	(255,215,0)
	dark golden rod	#B8860B	(184,134,11)
	golden rod	#DAA520	(218,165,32)
	pale golden rod	#EEE8AA	(238,232,170)
	dark khaki	#BDB76B	(189,183,107)
	khaki	#F0E68C	(240,230,140)

	olive	#808000	(128,128,0)
	yellow	#FFFF00	(255,255,0)
	yellow green	#9ACD32	(154,205,50)
	dark olive green	#556B2F	(85,107,47)
	olive drab	#6B8E23	(107,142,35)
	lawn green	#7CFC00	(124,252,0)
	chart reuse	#7FFF00	(127,255,0)
	green yellow	#ADFF2F	(173,255,47)
	dark green	#006400	(0,100,0)
	green	#008000	(0,128,0)
	forest green	#228B22	(34,139,34)
	lime	#00FF00	(0,255,0)
	lime green	#32CD32	(50,205,50)
	light green	#90EE90	(144,238,144)
	pale green	#98FB98	(152,251,152)
	dark sea green	#8FBC8F	(143,188,143)
	medium spring green	#00FA9A	(0,250,154)
	spring green	#00FF7F	(0,255,127)
	sea green	#2E8B57	(46,139,87)
	medium aqua marine	#66CDAA	(102,205,170)
	medium sea green	#3CB371	(60,179,113)
	light sea green	#20B2AA	(32,178,170)
	dark slate gray	#2F4F4F	(47,79,79)
	teal	#008080	(0,128,128)
	dark cyan	#008B8B	(0,139,139)
	aqua	#00FFFF	(0,255,255)
	cyan	#00FFFF	(0,255,255)
	light cyan	#E0FFFF	(224,255,255)
	dark turquoise	#00CED1	(0,206,209)
	turquoise	#40E0D0	(64,224,208)
	medium turquoise	#48D1CC	(72,209,204)

	pale turquoise	#AFEEEE	(175,238,238)
	aqua marine	#7FFFD4	(127,255,212)
	powder blue	#B0E0E6	(176,224,230)
	cadet blue	#5F9EA0	(95,158,160)
	steel blue	#4682B4	(70,130,180)
	corn flower blue	#6495ED	(100,149,237)
	deep sky blue	#00BFFF	(0,191,255)
	dodger blue	#1E90FF	(30,144,255)
	light blue	#ADD8E6	(173,216,230)
	sky blue	#87CEEB	(135,206,235)
	light sky blue	#87CEFA	(135,206,250)
	midnight blue	#191970	(25,25,112)
	navy	#000080	(0,0,128)
	dark blue	#00008B	(0,0,139)
	medium blue	#0000CD	(0,0,205)
	blue	#0000FF	(0,0,255)
	royal blue	#4169E1	(65,105,225)
	blue violet	#8A2BE2	(138,43,226)
	indigo	#4B0082	(75,0,130)
	dark slate blue	#483D8B	(72,61,139)
	slate blue	#6A5ACD	(106,90,205)
	medium slate blue	#7B68EE	(123,104,238)
	medium purple	#9370DB	(147,112,219)
	dark magenta	#8B008B	(139,0,139)
	dark violet	#9400D3	(148,0,211)
	dark orchid	#9932CC	(153,50,204)
	medium orchid	#BA55D3	(186,85,211)
	purple	#800080	(128,0,128)
	thistle	#D8BFD8	(216,191,216)
	plum	#DDA0DD	(221,160,221)
	violet	#EE82EE	(238,130,238)

	magenta / fuchsia	#FF00FF	(255,0,255)
	orchid	#DA70D6	(218,112,214)
	medium violet red	#C71585	(199,21,133)
	pale violet red	#DB7093	(219,112,147)
	deep pink	#FF1493	(255,20,147)
	hot pink	#FF69B4	(255,105,180)
	light pink	#FFB6C1	(255,182,193)
	pink	#FFC0CB	(255,192,203)
	antique white	#FAEBD7	(250,235,215)
	beige	#F5F5DC	(245,245,220)
	bisque	#FFE4C4	(255,228,196)
	blanched almond	#FFEBCD	(255,235,205)
	wheat	#F5DEB3	(245,222,179)
	corn silk	#FFF8DC	(255,248,220)
	lemon chiffon	#FFFACD	(255,250,205)
	light golden rod yellow	#FAFAD2	(250,250,210)
	light yellow	#FFFFE0	(255,255,224)
	saddle brown	#8B4513	(139,69,19)
	sienna	#A0522D	(160,82,45)
	chocolate	#D2691E	(210,105,30)
	peru	#CD853F	(205,133,63)
	sandy brown	#F4A460	(244,164,96)
	burly wood	#DEB887	(222,184,135)
	tan	#D2B48C	(210,180,140)
	rosy brown	#BC8F8F	(188,143,143)
	moccasin	#FFE4B5	(255,228,181)
	navajo white	#FFDEAD	(255,222,173)
	peach puff	#FFDAB9	(255,218,185)
	misty rose	#FFE4E1	(255,228,225)
	lavender blush	#FFF0F5	(255,240,245)
	linen	#FAF0E6	(250,240,230)

	old lace	#FDF5E6	(253,245,230)
	papaya whip	#FFEFD5	(255,239,213)
	sea shell	#FFF5EE	(255,245,238)
	mint cream	#F5FFFA	(245,255,250)
	slate gray	#708090	(112,128,144)
	light slate gray	#778899	(119,136,153)
	light steel blue	#B0C4DE	(176,196,222)
	lavender	#E6E6FA	(230,230,250)
	floral white	#FFFAF0	(255,250,240)
	alice blue	#F0F8FF	(240,248,255)
	ghost white	#F8F8FF	(248,248,255)
	honeydew	#F0FFF0	(240,255,240)
	ivory	#FFFFF0	(255,255,240)
	azure	#F0FFFF	(240,255,255)
	snow	#FFFAFA	(255,250,250)
	black	#000000	(0,0,0)
	dim gray / dim grey	#696969	(105,105,105)
	gray / grey	#808080	(128,128,128)
	dark gray / dark grey	#A9A9A9	(169,169,169)
	silver	#C0C0C0	(192,192,192)
	light gray / light grey	#D3D3D3	(211,211,211)
	gainsboro	#DCDCDC	(220,220,220)
	white smoke	#F5F5F5	(245,245,245)
	white	#FFFFFF	(255,255,255)

Manual Change History:

Date	Revision	Change
1/17/2014	REV1.0	Initial Version of this manual
3/24/204	REV1.1	Change Flash Memory map.
4/26/2014	REV1.2	Add Button_Label and Get_uiMessageAdvace GUI Commands.

Hardware Limited Warranty

Haidar Technology, LLC. Warrants its hardware products to be free from manufacturing defects in materials and workmanship under normal use for a period of one (1) year from the date of purchase from HAIDAR. This warranty extends to products purchased directly from HAIDAR or an authorized HAIDAR distributor. Purchasers should inquire of the distributor regarding the nature and extent of the distributor's warranty, if any. HAIDAR shall not be liable to honor the terms of this warranty if the product has been used in any application other than that for which it was intended, or if it has been subjected to misuse, accidental damage, modification, or improper installation procedures. Furthermore, this warranty does not cover any product that has had the serial number altered, defaced, or removed. This warranty shall be the sole and exclusive remedy to the original purchaser. In no event shall HAIDAR be liable for incidental or consequential damages of any kind (property or economic damages inclusive) arising from the sale or use of this equipment. HAIDAR is not liable for any claim made by a third party or made by the purchaser for a third party. HAIDAR shall, at its option, repair or replace any product found defective, without charge for parts or labor. Repaired or replaced equipment and parts supplied under this warranty shall be covered only by the unexpired portion of the warranty. Except as expressly set forth in this warranty, HAIDAR makes no other warranties, expressed or implied, nor authorizes any other party to offer any warranty, including any implied warranties of merchantability or fitness for a particular purpose. Any implied warranties that may be imposed by law are limited to the terms of this limited warranty. This warranty statement supercedes all previous warranties, and covers only the HAIDAR hardware.

Returns and Repair Policy

No merchandise may be returned for credit, exchange, or service without prior authorization from. To obtain warranty service, contact the factory and request an RMA (Return Merchandise Authorization) number. Enclose a note specifying the nature of the problem, name and phone number of contact person, RMA number, and return address. Authorized returns must be shipped freight prepaid to HAIDAR Technology LLC. with the RMA number clearly marked on the outside of all cartons. Shipments arriving freight collect or without an RMA number shall be subject to refusal. HAIDAR reserves the right in its sole and absolute discretion to charge a 15% restocking fee, plus shipping costs, on any products returned with an RMA.

Return freight charges following repair of items under warranty shall be paid by HAIDAR, shipping by standard ground carrier. In the event repairs are found to be non-warranty, return freight costs shall be paid by the purchaser.