GTT Protocol

For all variants of the GTT29A, GTT35A, GTT38A, GTT43A, GTT50A, and GTT70A

Protocol Manual

Revision 2.5

Firmware Revision: 2.0 or Higher

Revision History

Revision	Date	Description	Author
2.5	1 September 2016	Added Filled Slider and corrected Get Toggle State command	Divino
2.4	25 July 2016	Restructuring Manual for Firmware 2.5 Release	Divino
2.3	18 March 2016	Added Read Screen, Toggle, Slider, and Label Features	Clark
2.2	21 October 2014	Added Scripting, Label, and Strip Chart Features	Clark
2.1	8 April 2014	Added Scripting, Label, and Trace Features	Clark
2.0	8 October 2013	Initial Release	Clark

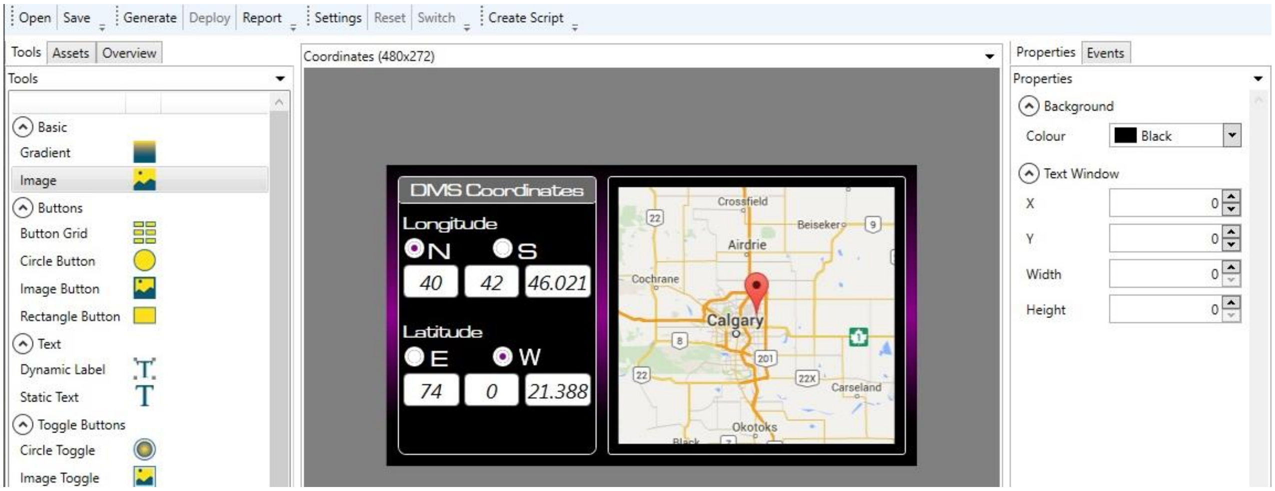
Contents

1 Introduction

1.1 Design

Design Tool

The GTT Design software, available at http://matrixorbital.ca/gtt-software, makes development for the GTT quick and easy, while still maintaining beautiful user interfaces and menu structures. Simulating the GTT display that is being used, the GTT Design Software allows users to place buttons, shapes, images, graphs and text exactly where they want on screen. With its intuitive design, users will be able to create and deploy multiple screens to be used on their GTT.



Background
Colour: Black

Text Window
X: 0
Y: 0
Width: 0
Height: 0

Figure 1: GTT Design Software

As shown above, the simple and intuitive design of the GTT Design tool provides users the ability to drag objects onto their simulated GTT screen. Once placed on screen, the user will be able to re-size, and re-position the object to their liking. The properties tab offers more precision when making adjustments to an object, and allows a user to change a plethora of other parameters of the object selected.

Loaded with the full library of GTT series commands, the GTT Design Software also provides users the ability to create scripts that can be loaded on the GTT. Furthermore, scripts can be linked to buttons created by the designer, and can be executed whenever the button is pressed.

Connections

In order to communicate with the GTT, you will need to connect to the appropriate communication header. From the factory, a standard GTT unit will come with RS232 selected as the communication protocol. If a different communication protocol is preferred over the standard RS232 protocol, the protocol select jumpers will have to be relocated in order to allow proper communication to the GTT. Header locations for each available protocol can be found in your respective GTT Hardware manual. GTT hardware manuals can be found at http://matrixorbital.ca/manuals/gtt-series.

Power can be applied to the GTT through the selected communication header, the alternate power connector, or if available, a power jack adapter on the display. Please consult your respective GTT Hardware manual for the power specifications, and power supply headers available on your unit.

Once the GTT is connected and powered, commands can be sent directly to the display using a terminal program, or one of the many communication tools we offer on our website. Communication settings may need to be adjusted before communication can proceed. It should also be noted that if the communication protocol has changed, the communication channel may need to be set, otherwise return messages from user inputs will not be received.

SD Card

The GTT includes a removable FAT16 format SD card with a 2GB capacity. The GTT is also capable of using higher capacity FAT32 SDHC and exFAT SDXC cards.

Communication

The multiple communication protocols available and simple command structure of the GTT means that a variety of applications can be used to communicate with the display. Basic default settings for serial protocols, which include USB, TTL, RS232, and RS422, as well as I^2C protocol are shown below.

Table 1: Serial Communication Settings

Speed	Data Bits	Parity	Stop Bits	Flow Control
115.2Kbps	8	None	1	RTS/CTS

Table 2: I^2C Communication Settings

Write	Read	Speed
80_d	81_d	Up to 100Kbps

Flow Control

The GTT comes with flow control functionality, allowing the data transmission rate to be managed when congestion occurs between the host and the GTT. Both Hardware, and Software flow control options are available on the GTT, and the user can select either type of flow control, based on the requirements for their project.

Hardware flow control makes use of the RTS and CTS pins available on the 6 pin Serial Communication header. Software flow control follows XON XOFF protocol to control data transmission. GTT's are capable of queuing 4096 bytes of unprocessed data within their data buffers. With flow control on, if the data buffer fills to the point where only 1 byte of space is free, the GTT will return a message to the host, and transmission of data will slow down, until more space frees up.

1.2 Basic Features

Text

By default, all bytes sent to the display are printed using the default font and standard ASCII encoding. For example, if the user sends characters 72, 101, 108, 108, and 111 to the display, “Hello” will be written on screen.

Commands

When the display detects the command prefix character 254, it will enter a command processing state and await the command number and its parameters. Multiple bytes are transferred in Big Endian format. Once the command is finished, the display will automatically return to displaying all bytes sent.

Table 3: Get Module ID bytes

Prefix	**254**	The command prefix
Message ID	**55**	Message ID 55, Get Module ID

If a user sends values 254 55, the display will process the 254 and enter a command processing state. Once the GTT processes the 55, it will recognize that a Get Module Type command has been received, and will access the onboard memory to identify itself. Once identified, the display will return the Module ID to the host. After the GTT returned its module ID, the display will return to its normal state and wait for more data.

Return Messages

When the display must return data to the host, it will use a standard message format. Each message will begin with the return message prefix 252, followed by the command number generating the message, a short value containing the length of the data in the message, followed by the data in the message.

Table 4: Example Return Message

Prefix	**252**	The return message prefix
Message ID	**55**	Message ID 55, Get Module ID
Length	**0**	Length MSB
	2	Length LSB
Data	**Byte[1]**	Module ID MSB
	Byte[2]	Module ID LSB

The sample above shows the expected return values from the Get Module ID command. In this manual, expected return messages are described below any required parameters.

Control Characters

In addition to text, the module will respond to a few of the default ASCII control characters while in the default mode. The display can be toggled between Windows and UNIX compatibility modes using the Control Character Mode command.

Table 5: Control Characters

	UNIX Compatibility Mode	Windows Compatibility Mode
7	The bell character will signal the Default Beep	The bell character will signal the Default Beep
10	Move the text insertion point to the beginning of the next line down	Move the text insertion point down one line
13	Move the text insertion point to the beginning of the next line down	Move the text insertion point back to the beginning of the current line

Drawing

The most basic commands available for the GTT line are the drawing features. Simple shapes, from pixels to triangles, can easily be drawn on the unit using a number of available commands. It should be noted that the coordinate system of the GTT references the top left pixel as 0,0 and increments positively to the right and down, as shown below.

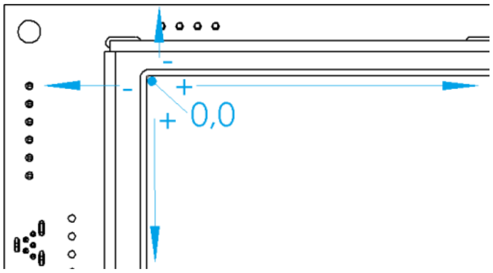


Figure 2: Pixel Coordinate Orientation (GTT43A shown)

The drawing colour can be set globally by specifying values for red, green, and blue channels, and will default to white. The Get Display Metrics function will report the number of bits available for each colour channel. The GTT will use the highest bits of any colour specified, dropping the lowest if necessary. For example, if the display uses 5 bits for red, setting the drawing colour to any value between 0 and 7 will result in the same, black, colour.

Buffers

Certain assets must be loaded into a buffer before they can be used by the GTT. Assets such as fonts, bitmaps, 9-Slice images, and animation files stored on the SD card must be loaded into a buffer before they can be rendered. When assets are loaded into a buffer, they are given an index number. While they are loaded in their buffer, they can be accessed at any time using the index number provided. Buffers can also be cleared at any time in order to free up buffer space.

Larger asset files will take time to load into their buffers. For example, animation files with a large quantity of frames will take a long time to load, and may cause long delays prior to each screen transition.

Depending on your project, it may be more beneficial to load all buffer files during your program initialization. This will lead to quicker response times during operation, in exchange for a longer initialization time. In contrast, you can load each buffer prior to each screen. This may result in slower response times between screens, but will provide balance between initialization and operation. It will be up to you to balance your buffer loading times accordingly, to meet your project needs.

Index Numbers

Certain assets require an index number to be provided during creation. Information about the asset that has been created is stored with these ID numbers. The user will be able to reference the asset ID numbers in future commands allowing the user to update or change the asset as their program progresses.

Assets that require an index number will fall under a specific category: Label, font, bitmap, 9-Slice, animation, bargraph, trace, keys, or region. Each category has 256 unique index slots available, meaning each category can have 256 unique assets at once. Furthermore, assets that fall under the same category must have different ID numbers otherwise they will not function properly.

For example when a new slider is created, all the parameters specified during creation are saved along with the ID number appointed to the slider. These parameters include the slider's coordinates on the screen, value range, track/button width and height, 9-Slice files, and style. If you want to retrieve the current value of the slider, you would use the Get Slider Value command, and specify the slider's ID number. This is helpful when multiple sliders are on screen at the same time.

Assets that fall under the same category will share the category's index number pool. For example, touch regions, toggles, and sliders all fall under the regions category, and have access to the same index numbers. This is important to note, because in order for an asset to function properly, it will need to have its own ID number, unique to the rest of the assets in their category. In the case that a slider and a toggle are given the same ID, both assets will be visible on screen, but only one will be functional. This is due to one asset overwriting the data of the other asset.

Fonts

Fonts can be uploaded to the SD memory card and buffered for use on the display. If no other font has been selected or loaded, the GTT will default to a non-scalable proggy font when updating the screen with text.

Bitmaps

Bitmaps are uploaded to the SD memory card before use. They can also be used to create touch regions or animations. Furthermore, a specific colour can be specified to appear transparent when the bitmap is rendered.

Bargraphs

Bargraphs simplify the display of multiple bargraphs on the screen by taking care of the calculations and placement of graphics. Once a bargraph is created, only the new value needs to be sent to update it. The ratio of the new value to the minimum and maximum levels is automatically calculated, and the graphic is updated. Bargraph information is stored in a series of bargraph buffers. The index is chosen when the bargraph is created, and used to reference the bargraph in future commands.

Traces

Traces provide an easy method to display a stream of information on screen. Once a trace is created, the user will only have to worry about updating the trace with new data, as the GTT will manage the calculations, placement of graphics, and shifting of data. Upon creation, Traces are given an index number, which will be used to reference the trace in future commands. Trace information is stored in a series of trace specific buffers.

1.3 Advanced Features

9-Slice

9-Slices files can modify and scale a bitmap without distorting its geometry. A 9-Slice file will cut a bitmap into 9 separate pieces, and automatically adjust each of those 9 pieces in order to scale an image. For instructions on creating a 9-Slice file, along with an example of a 9-Slice, see section 3.2 of this manual

9-Slice Graphs

9-Slice graphs offer similar functionality to standard bargraphs, but allow for complex graphics to be used for more detailed rendering. With 9-Slice bargraphs, the GTT will take care of the calculations necessary when new information is received, and update the graphics appropriately. The 9-Slice bargraph allows a user to specify 9-Slice images loaded in the GTT's buffer, and will use those images when rendering the graph. 9-Slice bargraph information is stored in a shared buffer with standard bargraphs.

Animations

Animation files may be saved and accessed from the SD card, and can be played on the GTT. In order to run an animation on the display, an animation text file, and all accompanying animation frame images must be saved on the GTT SD card. Details on how to create an animation text file can be found in section 3.2 of this manual

Keypad

Unique values can be assigned for up to 25 keys. When a key event occurs it will be saved to a 20 key buffer. Key events will generate a return message that can be transmitted immediately or polled by toggling the auto transmit key press command. A sample return message is shown below.

Table 6: Example Keypad Response

Prefix	**252**	The return message prefix
Message ID	**55**	Message ID 55, Get Module ID
Length	**0**	Length MSB
	2	Length LSB
Event	**0**	Key Event
Key ID	**65**	ID of key pressed

Key presses will have a message ID of 165, and a data length equal to the number of bytes currently in the key buffer. Each key value will be preceded by an event byte as per the Keypad Event Types table.

Table 7: Keypad Event Types

Value	0	1	2
Event	Press	Release	Repeat

Touch

Touch input allows the GTT to return various types of up, down and move messages depending on the reporting style. Two distinct styles are available: region and coordinate. Both generate a return message with an identification number of 135, followed by event information.

Table 8: Touch Event Types

Value	0	1	2
Event	Down	Up	Move

In coordinate mode, the GTT will send an event type as listed above followed by signed short x and y coordinates of the touch location.

Table 9: Example Co-ordinate Response

Prefix	252	The return message prefix
Message ID	135	Message ID 135, Touch Input
Length	0	Length MSB
	5	Length LSB
Event	0	Touch Event type
X	0	MSB of X coordinate
	50	LSB of X coordinate
Y	0	MSB of Y coordinate
	10	LSB of Y coordinate

In region mode, rectangular buttons are defined on the screen. When touch activities occur within regions, a visual update accompanies the event report listed in the Touch Event Types table. Events that occur outside defined regions may be reported as Region 255, when reporting is turned on.

Table 10: Example Region Response

Prefix	252	The return message prefix
Message ID	135	Message ID 135, Touch Input
Length	0	Length MSB
	2	Length LSB
Event	0	Touch Event
Region	5	ID of region pressed

Region

Touch regions may be defined using a simple text file for speed and greater ease of use. In cases where multiple screens share the same region placement, it may be beneficial to run a region script before each screen, rather than have your program create each region individually. Details on creating a new region file can be found at the end of this manual.

Scripts

Script files can be created and loaded onto the GTT, then executed anytime during field operation. A script file is comprised of a list of commands that a user wants to run, along with their corresponding parameter information. When a script is executed, all the commands and data within that script will be parsed as if it came from the input communications port. This allows a user to execute multiple GTT commands by only sending one Run Script command to the GTT. Details on creating and running scripts on the GTT can be found in section 3.2 of this manual.

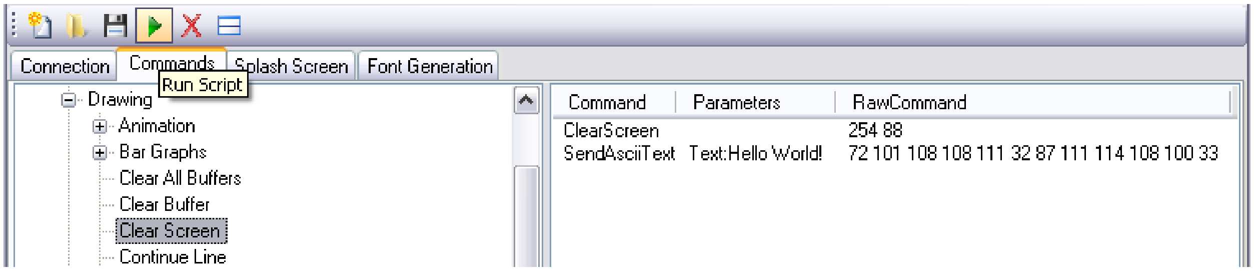
Autoexec

On startup, the GTT will check the root directory of the SD card for a file named AUTOEXEC. If that file exists, it will be loaded directly into the in buffer and parsed as if it came from the input communications port. This is useful for having custom power on defaults. Details on setting up an autoexec for your project, along with an example Autoexec file, can be found in section 3.2 of this manual.

1.4 Support

Support Tool

Downloaded from http://www.matrixorbital.ca/software/, the GTT Project support tool provides a simple graphic interface with the full library of GTT series commands. This program allows users to drag and drop commands into a list that can be transmitted to the GTT, saved, and even loaded for later use.



Command	Parameters	RawCommand
ClearScreen		254 88
SendAsciiText	Text:Hello World!	72 101 108 108 111 32 87 111 114 108 100 33

Figure 3: GTT Support Tools

As you can see, each command added is displayed its name, any applicable parameters, and finally, a byte by byte account of the information that will be sent in decimal notation. While this list of commands can be saved and recalled later, it can also be converted into a binary file using the save as feature. This will allow easy creation of AUTOEXEC startup files, and integration into application specific code. Finally, the support tool provides a debug window that will display the information flow to and from your GTT to ensure your command list executes exactly as it was envisioned.

Application Notes

Full demonstration programs and code are available for many different Matrix Orbital displays in a number of different languages from the Application Note section at www.matrixorbital.ca/appnotes.

In addition, all files required to run the short examples described in the Advanced Features section are available for download from www.matrixorbital.ca/manuals/GTT_Series. Each example runs as an autoexec script and is described in the Instructions document.

Finally, a self-contained demo highlighting many of the features available in the GTT line is available at http://www.matrixorbital.ca/gtt-rev2-0-feature-demo. No code is required as all functionality is provided through scripts. Simply copy the required files to your GTT to run the interactive demo.

For additional information regarding the features implemented, please see the Commands section below. If you have any questions please don't hesitate to contact a knowledgeable Matrix Orbital technical support representative.

Firmware Upgrades

After release, Matrix Orbital may publish updates to the GTT code base or functionality that can be easily applied to the unit in the field. While in mass storage mode replace all of the files in the GTT upgrade folder with the latest package available from www.matrixorbital.ca/software/GTT/. Then, cycle power to the unit, wait for the upgrade to complete, and allow the screen to reboot. Finally, replace the GTT in your application and enjoy the new additions to the display you've come to know and love.

2 Commands

2.1 Communication

1.1 Enter Mass Storage Mode	Dec	254 4		2.0
	Hex	FE 04		
	ASCII	þ [EOT]		

Programmatically force the GTT to enter mass storage mode.

1.2 Set Communication Channel	Dec	254 5	Channel	2.0
	Hex	FE 05	Channel	
	ASCII	þ [ENQ]	Channel	

Set the default communication channel to be used for asynchronous data transmission. Asynchronous data includes responses from the keypad and touchpad. Synchronous data requests, such as commands, are always answered on the requesting channel.

Channel	**Byte**	Communication channel type, as per eChannel Values.

Table 11: eChannel Values

Value	Description
0	None
1	Serial
2	I2C
3	USB
4	CAN
5	SPI
255	Current

1.3 Set Baud Rate	Dec	254 57	BaudRate	2.0
	Hex	FE 39	BaudRate	
	ASCII	þ 9	BaudRate	

Set the serial data rate used by the GTT. The change is implemented immediately after the last parameter byte has been received. Baud rate will reset to 115,200 on power up unless otherwise defined in the autoexec file. This is a serial command only.

BaudRate	**Integer**	The desired baud rate value.

1.4 Set Flow Control Mode	Dec	254 58	FlowControl	2.0
	Hex	FE 3A	FlowControl	
	ASCII	þ :	FlowControl	

Set the hardware flow control mode used by the GTT. The default, and recommended, setting is RTSCTS. If buffer overflow is observed please ensure hardware flow control is set to RTSCTS, and implemented. This is a serial command only.

FlowControl	**Byte**	Flow control setting, as per eFlowControl Values.

Table 12: eFlowControl Values

Value	Description
0	Off
1	RTSCTS

1.5 Set I2C Address	Dec	254 247	I2Caddress	2.0
	Hex	FE F7	I2Caddress	
	ASCII	þ ÷	I2Caddress	

Set the I2C write address of the GTT. Only even values are permitted as the next odd address will become the read address. Default 8 bit address on startup is 80 decimal (0x50 hex) unless otherwise defined in the I2C.cfg file in the \system folder, or the autoexec file. This is an I2C command only.

I2Caddress	**Byte**	I2C write address, must be an even value.

1.6 Echo	Dec	254 255	Message	2.0
	Hex	FE FF	Message	
	ASCII	þ ÿ	Message	

Ask the GTT to echo a string that is sent to it. This command can be used to test communication or indicate completion of a successful power up when placed in the autoexec file.

Message	**ASCII String**	An arbitrary string that the module will return. Limited to 4KB in length.
Return Message	**252 255 Length**	ReturnMessage
ReturnMessage	**ASCII String**	The same arbitrary string originally sent.

2.2 Module

2.1 Get Protocol Revision	Dec	254 0	2.0
	Hex	FE 00	
	ASCII	þ [NUL]	

Get the firmware version currently installed on the GTT. Minor revisions will indicate an addition only, while major revisions will alter or remove commands; consult the appropriate changelog for more information on changes. For each command in this manual, the minimum firmware version required is listed at the top right.

Return Message	**252 0 Length**	Major Minor
Major	**Byte**	Major revision of the protocol used.
Minor	**Byte**	Minor revision of the protocol used.

2.2 Reset Module	Dec	254 1	2.0
	Hex	FE 01	
	ASCII	þ [SOH]	

Initiate a soft reset of the GTT. The standard start up sequence will ensue and all settings will revert to defaults.

2.3 Delay	Dec	254 2	Time	2.0
	Hex	FE 02	Time	
	ASCII	þ [STX]	Time	

Pause command execution to and responses from the GTT for the specified length of time.

Time	**Short**	Length of delay in milliseconds.

2.4 Get Display Metrics	Dec	254 3	2.0
	Hex	FE 03	
	ASCII	þ [ETX]	

Get the width, height, and colour resolution of the GTT screen.

Return Message	252 3 Length	Width Height BitsRed BitsGreen BitsBlue
Width	Short	The width of the current display resolution in pixels.
Height	Short	The height of the current display resolution in pixels.
BitsRed	Byte	The number of bits used in the red channel. When less than 8 bits, byte length colour commands use the highest bits.
BitsGreen	Byte	The number of bits used in the green channel. When less than 8 bits, byte length colour commands use the highest bits.
BitsBlue	Byte	The number of bits used in the blue channel. When less than 8 bits, byte length colour commands use the highest bits.

2.5 Set Screen Orientation	Dec	254 50	Orientation	2.5
	Hex	FE 32	Orientation	
	ASCII	þ 2	Orientation	

Set the orientation of the GTT screen. This command is useful for applications where the GTT is installed in a portrait or flipped orientation. Default is Landscape.

Orientation	Byte	Desired screen orientation, as per ePanelOrientation Values.

Table 13: ePanelOrientation Values

Value	Description
0	Landscape
1	PortraitClockwise
2	LandscapeFlipped
3	PortraitCounterClockwise

2.6 Set Customer Data	Dec	254 52	Length Data	2.0
	Hex	FE 34	Length Data	
	ASCII	þ 4	Length Data	

Write information to a specific file in non-volatile. Up to 255 bytes can be written to the userdata.dat file in the \system folder of the GTT SD card using this command. This data could potentially be unit identification, network information, system settings, or anything else specific to the module.

Length	Byte	Length of the data to be transferred, in bytes.
Data	Byte(s)	Data to be written to the SD card.

2.7 Get Customer Data	Dec	254 53	2.0
	Hex	FE 35	
	ASCII	þ 5	

Read data from the userdata.dat file in the \system folder of the GTT SD card.

Return Message	252 53 Length	Length Data
Length	Byte	Length of the data to be transferred, in bytes.
Data	Byte(s)	Data read from the SD Card.

2.8 Get Module Type	Dec	254 55	2.0
	Hex	FE 37	
	ASCII	þ 7	

Get a two byte value used to identify the GTT.

Return Message	**252 55 Length**	**Module**
Module	**Short**	The unique number of the module, as per eModule Values.

Table 14: eModule Values

Value	Description
37638	GTT35A
37648	GTT38A
37633	GTT43A
37634	GTT50A
37635	GTT57A
37636	GTT70A

2.9 Get Module String	Dec	254 56	2.0
	Hex	FE 38	
	ASCII	þ 8	

Get a string value used to identify the GTT.

Return Message	**252 56 Length**	**ModuleString**
ModuleString	**ASCII String**	The name of the module.

2.10 Set Backlight Brightness	Dec	254 153	Brightness	2.0
	Hex	FE 99	Brightness	
	ASCII	þ ™	Brightness	

Set the brightness of the display backlight. This setting is not saved to memory, but may be included in the autoexec file.

Brightness	**Byte**	The backlight brightness, a value between 0 (off) and 255 (maximum).

2.11 Get Backlight Brightness	Dec	254 154	2.0
	Hex	FE 9A	
	ASCII	þ š	

Get the current display backlight brightness setting.

Return Message	**252 154 Length**	**Brightness**
Brightness	**Byte**	The current backlight brightness.

2.12 Write ScratchPad	Dec	254 204	Index Length Data	2.0
	Hex	FE CC	Index Length Data	
	ASCII	þ Ì	Index Length Data	

Write information to volatile memory for temporary storage during operation. A total of 512 bytes is reserved for the scratch pad in GTT RAM.

Index	**Short**	Starting index of the data to be written.
Length	**Short**	Length of the data to be transferred, in bytes.
Data	**Byte(s)**	Data to temporarily save in volatile memory.

2.13 Read ScratchPad	Dec	254 205	Index Size	2.0
	Hex	FE CD	Index Size	
	ASCII	þ Í	Index Size	

Read information that was previously stored in volatile memory.

Index	**Short**	Starting index of the data to be read.
Size	**Short**	Length of the data requested.
Return Message	**252 205 Length**	Length Result
Length	**Short**	Length of the data to be transferred, in bytes.
Result	**Byte(s)**	Data read from specified location in volatile memory.

2.3 Drawing

3.1 Set Background Drawing Colour	Dec	254 86	R G B	2.0
	Hex	FE 56	R G B	
	ASCII	þ V	R G B	

Set the colour that is used for the background of all drawing commands, and fills the screen when a Clear Screen command is sent to the GTT. The default background colour is black.

R	**Byte**	Intensity of red, 0 to 255, limited to display metrics.
G	**Byte**	Intensity of green, 0 to 255, limited to display metrics.
B	**Byte**	Intensity of blue, 0 to 255, limited to display metrics.

3.2 Get Background Drawing Colour	Dec	254 87	2.0
	Hex	FE 57	
	ASCII	þ W	

Get the current background drawing colour of the GTT.

Return Message	**252 87 Length**	R G B
R	**Byte**	Intensity of red, 0 to 255, limited to display metrics.
G	**Byte**	Intensity of green, 0 to 255, limited to display metrics.
B	**Byte**	Intensity of blue, 0 to 255, limited to display metrics.

3.3 Clear Screen	Dec	254 88	2.0
	Hex	FE 58	
	ASCII	þ X	

Clear the screen, and reset the coordinates for both the Continue Line and Text Window commands to zero.

3.4 Scroll Screen	Dec	254 89	X Y Width Height MoveX MoveY	2.0
	Hex	FE 59	X Y Width Height MoveX MoveY	
	ASCII	þ Y	X Y Width Height MoveX MoveY	

Scroll the contents of a specified portion of the GTT screen.

X	Signed Short	Leftmost coordinate of the scroll window.
Y	Signed Short	Topmost coordinate of the scroll window.
Width	Signed Short	Width of the scroll window.
Height	Signed Short	Height of the scroll window.
MoveX	Signed Short	Number of pixels to scroll horizontally.
MoveY	Signed Short	Number of pixels to scroll vertically.

3.5 Enable Manual Update	Dec	254 90	Enable	2.0
	Hex	FE 5A	Enable	
	ASCII	þ Z	Enable	

Enable manual graphic updates. This command stops all drawing commands from automatically updating the screen and sends them to the display buffer to be executed simultaneously when the Manual Update command is sent to the GTT. This command is useful for displaying a complicated image as a single visual update. Default is disabled.

Enable	Byte	Desired manual update setting, as per eEnable Values.

Table 15: eEnable Values

Value	Description
0	Disable
1	Enable

3.6 Manual Update	Dec	254 91	2.0
	Hex	FE 5B	
	ASCII	þ [	

Immediately push all contents of the display buffer to the screen. This command has no effect if manual update is disabled.

3.7 Flush Region	Dec	254 92	X Y Width Height	2.0
	Hex	FE 5C	X Y Width Height	
	ASCII	þ \	X Y Width Height	

Immediately push all graphic data in a specified region of the display buffer to the screen. This command has no effect if manual update is disabled.

X	Signed Short	Leftmost coordinate of the flush window.
Y	Signed Short	Topmost coordinate of the flush window.
Width	Signed Short	Width of the flush window.
Height	Signed Short	Height of the flush window.

3.8 Set Drawing Colour	Dec	254 99	R G B	2.0
	Hex	FE 63	R G B	
	ASCII	þ c	R G B	

Set the colour that is used for the foreground of all drawing commands sent to the GTT. The default drawing colour is white.

R	Byte	Intensity of red, 0 to 255, limited to display metrics.
G	Byte	Intensity of green, 0 to 255, limited to display metrics.
B	Byte	Intensity of blue, 0 to 255, limited to display metrics.

3.9 Get Drawing Colour	Dec	254 100	2.0
	Hex	FE 64	
	ASCII	þ d	

Get the current foreground drawing colour of the GTT.

Return Message	252 100 Length	R G B
R	Byte	Intensity of red, 0 to 255, limited to display metrics.
G	Byte	Intensity of green, 0 to 255, limited to display metrics.
B	Byte	Intensity of blue, 0 to 255, limited to display metrics.

3.10 Continue Line	Dec	254 101	X Y	2.0
	Hex	FE 65	X Y	
	ASCII	þ e	X Y	

Draw a line from the last point drawn to the coordinate specified using the current drawing colour. The last stored point is automatically updated from Draw Pixel, Draw Line, and Continue Line commands.

X	Signed Short	Horizontal coordinate of line terminus.
Y	Signed Short	Vertical coordinate of line terminus.

3.11 Draw Line	Dec	254 108	X1 Y1 X2 Y2	2.0
	Hex	FE 6C	X1 Y1 X2 Y2	
	ASCII	þ l	X1 Y1 X2 Y2	

Draw a line connecting two termini using the current drawing colour. Lines may be rendered differently when drawn right to left versus left to right.

X1	Signed Short	Horizontal coordinate of first line terminus.
Y1	Signed Short	Vertical coordinate of first line terminus.
X2	Signed Short	Horizontal coordinate of second line terminus.
Y2	Signed Short	Vertical coordinate of second line terminus.

3.12 Draw Pixel	Dec	254 112	X Y	2.0
	Hex	FE 70	X Y	
	ASCII	þ p	X Y	

Draw a single pixel at the specified coordinate using the current drawing colour.

X	Signed Short	Horizontal position of pixel to be drawn.
Y	Signed Short	Vertical position of pixel to be drawn.

3.13 Draw Rectangle	Dec	254 114	X Y Width Height	2.0
	Hex	FE 72	X Y Width Height	
	ASCII	þ r	X Y Width Height	

Draw a rectangular frame one pixel wide using the current drawing colour.

X	**Signed Short**	Leftmost coordinate of the rectangle.
Y	**Signed Short**	Topmost coordinate of the rectangle.
Width	**Short**	Width of the rectangle.
Height	**Short**	Height of the rectangle.

3.14 Draw Filled Rectangle	Dec	254 120	X Y Width Height	2.0
	Hex	FE 78	X Y Width Height	
	ASCII	þ x	X Y Width Height	

Draw a filled rectangle using the current drawing colour.

X	**Signed Short**	Leftmost coordinate of the rectangle.
Y	**Signed Short**	Topmost coordinate of the rectangle.
Width	**Short**	Width of the rectangle.
Height	**Short**	Height of the rectangle.

3.15 Draw Circle	Dec	254 123	X Y Radius	2.0
	Hex	FE 7B	X Y Radius	
	ASCII	þ {	X Y Radius	

Draw a circular frame one pixel wide using the current drawing colour.

X	**Signed Short**	Horizontal coordinate of circle centre.
Y	**Signed Short**	Vertical coordinate of circle centre.
Radius	**Short**	Radius of the circle.

3.16 Draw Filled Circle	Dec	254 124	X Y Radius	2.0
	Hex	FE 7C	X Y Radius	
	ASCII	þ |	X Y Radius	

Draw a filled circle using the current drawing colour.

X	**Signed Short**	Horizontal coordinate of circle centre.
Y	**Signed Short**	Vertical coordinate of circle centre.
Radius	**Short**	Radius of the circle.

3.17 Draw an Ellipse	Dec	254 125	X Y XRadius YRadius	2.0
	Hex	FE 7D	X Y XRadius YRadius	
	ASCII	þ }	X Y XRadius YRadius	

Draw an elliptical frame one pixel wide using the current drawing colour.

X	**Signed Short**	Horizontal coordinate of ellipse centre.
Y	**Signed Short**	Vertical coordinate of ellipse centre.
XRadius	**Short**	Horizontal Radius of the ellipse.
YRadius	**Short**	Vertical Radius of the ellipse.

3.18 Draw a Filled Ellipse	Dec	254 126	X Y XRadius YRadius	2.0
	Hex	FE 7E	X Y XRadius YRadius	
	ASCII	þ ~	X Y XRadius YRadius	

Draw a filled ellipse using the current drawing colour.

X	Signed Short	Horizontal coordinate of ellipse centre.
Y	Signed Short	Vertical coordinate of ellipse centre.
XRadius	Short	Horizontal Radius of the ellipse.
YRadius	Short	Vertical Radius of the ellipse.

3.19 Draw Rounded Rectangle	Dec	254 127	X Y Width Height Radius	2.0
	Hex	FE 7F	X Y Width Height Radius	
	ASCII	þ □	X Y Width Height Radius	

Draw a rectangular frame one pixel wide with rounded corners using the current drawing colour. The radius must be equal to or less than half the length of the smallest side of the rectangle.

X	Signed Short	Leftmost coordinate of the rectangle.
Y	Signed Short	Topmost coordinate of the rectangle.
Width	Signed Short	Width of the rectangle.
Height	Signed Short	Height of the rectangle.
Radius	Short	Radius of the rounded corners.

3.20 Draw Filled Rounded Rectangle	Dec	254 128	X Y Width Height Radius	2.0
	Hex	FE 80	X Y Width Height Radius	
	ASCII	þ €	X Y Width Height Radius	

Draw a filled rectangle with rounded corners using the current drawing colour. The radius must be equal to or less than half the length of the smallest side of the rectangle.

X	Signed Short	Leftmost coordinate of the rectangle.
Y	Signed Short	Topmost coordinate of the rectangle.
Width	Signed Short	Width of the rectangle.
Height	Signed Short	Height of the rectangle.
Radius	Short	Radius of the rounded corners.

3.21 Draw Triangle	Dec	254 129	X1 Y1 X2 Y2 X3 Y3	2.0
	Hex	FE 81	X1 Y1 X2 Y2 X3 Y3	
	ASCII	þ •	X1 Y1 X2 Y2 X3 Y3	

Draw a triangular frame one pixel wide using the current drawing colour.

X1	Signed Short	Horizontal coordinate of the first point.
Y1	Signed Short	Vertical coordinate of the first point.
X2	Signed Short	Horizontal coordinate of the second point.
Y2	Signed Short	Vertical coordinate of the second point.
X3	Signed Short	Horizontal coordinate of the third point.
Y3	Signed Short	Vertical coordinate of the third point.

3.22 Draw Filled Triangle	Dec	254 130	X1 Y1 X2 Y2 X3 Y3	2.0
	Hex	FE 82	X1 Y1 X2 Y2 X3 Y3	
	ASCII	þ ,	X1 Y1 X2 Y2 X3 Y3	
Draw a filled triangle using the current drawing colour.				
X1	**Signed Short**	Horizontal coordinate of the first point.		
Y1	**Signed Short**	Vertical coordinate of the first point.		
X2	**Signed Short**	Horizontal coordinate of the second point.		
Y2	**Signed Short**	Vertical coordinate of the second point.		
X3	**Signed Short**	Horizontal coordinate of the third point.		
Y3	**Signed Short**	Vertical coordinate of the third point.		

2.4 Buffers

4.1 Load Font	Dec	254 40	FontID FileName	2.0
	Hex	FE 28	FontID FileName	
	ASCII	þ (	FontID FileName	
Load a .ttf or .otf file from the SD card into a font buffer for use. Support for.otf fonts was added at firmware version 2.5.				
FontID	**Byte**	Index used to identify the font. Specific to fonts.		
FileName	**ASCII String**	Filename, including path from the root folder, of the font file to load.		
Return Message	**252 40 Length**	**Result**		
Result	**Byte**	Outcome of Load Font command, as per eStatusCode Values.		

Table 16: eStatusCode Values

Value	Description
0	FileNotFound
1	InvalidBitmapFileFormat
2	Invalid9SliceMetrics
3	Invalid9SliceIndex
4	InvalidBitmapIndex
5	InvalidBargraphIndex
6	InvalidAnimationIndex
7	InvalidAnimationFileFormat
8	InvalidFontIndex
9	InvalidCommandParameters
10	DisplayisOUTofRAM
11	InvalidRegionFileFormat
12	InvalidTouchCalibration
13	SuccessfulTouchCalibration
14	InvalidFileFormat
15	InvalidTraceIndex
16	InvalidTouchRegion
17	InvalidLabelIndex
254	Success
255	UnknownException

4.2 Read Screen Rectangle	Dec	254 94	X Y Width Height Format	2.4
	Hex	FE 5E	X Y Width Height Format	
	ASCII	þ ^	X Y Width Height Format	

Read the current value of every pixel in the specified screen area. Three byte values, representing red, green, and blue colour levels are returned for every pixel. The specified area must be less than 21,845 pixels in area due to return message restrictions. Please note, it may take a considerable length of time to read large screen areas.

X	Short	Leftmost coordinate of the screen rectangle to read.
Y	Short	Topmost coordinate of the screen rectangle to read.
Width	Short	Width of the screen rectangle to read.
Height	Short	Height of the screen rectangle to read.
Format	Byte	Pixel format of the screen data, as per ePixelFormat Values.
Return Message	**252 94 Length**	**Result Format Length Data**
Result	Byte	Outcome of the Read Screen Rectangle command, as per eStatusCode Values.
Format	Byte	Pixel format of the screen data, as per ePixelFormat.
Length	Short	Length of the data to be transferred, in bytes.
Data	Byte(s)	Current pixel data for every point within the specific rectangle, as per ePixelFormat. Values start at the top left of the rectangle, moving right, then down.

Table 17: ePixelFormat Values

Value	Description
0	RGB16
1	RGB24
3	BGR24

4.3 Load Bitmap	Dec	254 95	BitmapID FileName	2.0
	Hex	FE 5F	BitmapID FileName	
	ASCII	þ _	BitmapID FileName	

Load a bitmap file from the SD card into a bitmap buffer for use. Supported formats are BMP, GIF, JPG, and PNG. All files should be in RGB format; alpha and other channels are not supported.

BitmapID	Byte	Index used to identify the bitmap. Specific to bitmaps, and screen rectangles.
FileName	ASCII String	Filename, and path from the root folder, of the bitmap file to load.
Return Message	**252 95 Length**	**Result**
Result	Byte	Outcome of Load Bitmap command, as per eStatusCode Values.

4.4 Copy Screen Rectangle	Dec	254 96	BitmapID X Y Width Height	2.0
	Hex	FE 60	BitmapID X Y Width Height	
	ASCII	þ `	BitmapID X Y Width Height	

Load a copy of a specific portion of the screen into a bitmap buffer for later use.

BitmapID	Byte	Index used to identify the screen section. Specific to bitmaps and screen rectangles.
X	Signed Short	Leftmost coordinate.
Y	Signed Short	Topmost coordinate.
Width	Short	Width of the screen section.
Height	Short	Height of the screen section.

4.5 Load 9-Slice	Dec	254 144	NineSliceID Filename	2.0
	Hex	FE 90	NineSliceID Filename	
	ASCII	þ •	NineSliceID Filename	
Load a 9-slice file from the SD card into a 9-Slice buffer for use. Refer to the 9-Slices entry in the Features section for more information.				
NineSliceID	Byte	Index used to identify the 9-slice. Specific to 9-slices.		
Filename	ASCII String	Filename, and path from the root folder, of the 9-Slice file to load.		
Return Message	**252 144 Length**	Result		
Result	Byte	Outcome of Load 9-Slice command, as per eStatusCode Values.		

4.6 Load Animation	Dec	254 192	AnimationID Filename	2.0
	Hex	FE C0	AnimationID Filename	
	ASCII	þ À	AnimationID Filename	
Load an animation file from the SD card into an animation buffer for use. Refer to the Animations entry in the Features section for more information.				
AnimationID	Byte	Index used to identify this animation file in the animation buffer.		
Filename	ASCII String	Filename, and path from the root folder, of the animation file to load.		

4.7 Clear a Buffer	Dec	254 208	Type ID	2.0
	Hex	FE D0	Type ID	
	ASCII	þ Ð	Type ID	
Clear data from a specific index of the selected buffer type to free RAM. Labels and Traces save a background image to a bitmap buffer upon initialization, and will be affected by this command.				
Type	Byte	Type of buffer to clear, as per eBuffers Values.		
ID	Byte	Index of the file to be cleared from buffer memory.		

Table 18: eBuffers Values

Value	Description
0	Animations
1	Bitmaps
2	NineSlices
3	Fonts
4	Labels
5	Traces

4.8 Clear All Buffers	Dec	254 209	2.0
	Hex	FE D1	
	ASCII	þ Ñ	
Clear all data from all buffers to free significant RAM.			

2.5 Text

5.1 Create a Label				
	Dec	254 16	LabelID X Y Width Height Rot VJst HJst Font R G B	2.1
	Hex	FE 10	LabelID X Y Width Height Rot VJst HJst Font R G B	
	ASCII	þ [DLE]	LabelID X Y Width Height Rot VJst HJst Font R G B	

Designate a portion of the screen that can be updated with one line of text. A label is useful for displaying dynamic strings or changing numeric variables. Please note that the background of the label is saved to RAM upon creation and redrawn before each update.

LabelID	Byte	Index used to identify this label in the label list.
X	Signed Short	Leftmost coordinate of the label region.
Y	Signed Short	Topmost coordinate of the label region.
Width	Signed Short	Width of the label region in pixels.
Height	Signed Short	Height of the label region in pixels.
Rot	Signed Short	Rotation of the text within the label.
VJst	Byte	Vertical justification of text within the label, as per eFontAlignVertical Values.
HJst	Byte	Horizontal justification of text within the label, as per eFontAlignHorizontal Values.
Font	Byte	Font index of a previously loaded font to be used for the label.
R	Byte	Intensity of red, 0 to 255, used for label font colour.
G	Byte	Intensity of green, 0 to 255, used for label font colour.
B	Byte	Intensity of blue, 0 to 255, used for label font colour.

Table 19: eFontAlignVertical Values

Value	Description
0	Top
1	Bottom
2	Center

Table 20: eFontAlignHorizontal Values

Value	Description
0	Left
1	Right
2	Center

5.2 Update a Label (UTF8)				
	Dec	254 17	LabelID Format Value	2.1
	Hex	FE 11	LabelID Format Value	
	ASCII	þ	LabelID Format Value	

Update a previously created label with new UTF8 text. Send a null character (empty string) to clear a label.

LabelID	Byte	Index used to identify this label in the label list.
Format	Fixed Decimal	Format of the string that will update the label. For UTF8 specify 2.
Value	UTF8 String	New UTF-8 string to display within the label. String should be a single line in height.

5.3 Update a Label (ASCII)	Dec	254 17	LabelID Format Value	2.1
	Hex	FE 11	LabelID Format Value	
	ASCII	þ	LabelID Format Value	

Update a previously created label with new ASCII text. Send a null character (empty string) to clear a label.

LabelID	**Byte**	Index used to identify this label in the label list.
Format	**Fixed Decimal**	Format of the ASCII string that will update the label. For ASCII specify 0.
Value	**ASCII String**	New ASCII formatted string to display within the label. String should be a single line in height.

5.4 Update a Label (Unicode)	Dec	254 17	LabelID Format Value	2.1
	Hex	FE 11	LabelID Format Value	
	ASCII	þ	LabelID Format Value	

Update a previously created label with new Unicode text. Send a null character (empty string) to clear a label.

LabelID	**Byte**	Index used to identify this label in the label list.
Format	**Fixed Decimal**	Format of the string that will update the label. For Unicode specify 1.
Value	**Unicode String**	New Unicode string to display within the label. String should be a single line in height.

5.5 Set Label Activation State	Dec	254 19	LabelID State	2.4
	Hex	FE 13	LabelID State	
	ASCII	þ	LabelID State	

Set the activation state of an existing label. This command can be used to temporarily disable updates from appearing on the screen, without deleting a label. Default after label creation is Active.

LabelID	**Byte**	Index used to identify this label in the label list.
State	**Byte**	New label activation state, as per eActivation Values.
Return Message	**252 19 Length**	**Result**
Result	**Byte**	Outcome of Set Label Activation command, as per eStatusCode Values.

Table 21: eActivation Values

Value	Description
0	Inactive
1	Active

5.6 Get Label Activation State	Dec	254 20	LabelID	2.4
	Hex	FE 14	LabelID	
	ASCII	þ	LabelID	

Get the current activation state of an existing label.

LabelID	**Byte**	Index used to identify this label in the label list.
Return Message	**252 20 Length**	**Result State**
Result	**Byte**	Outcome of Get Label Activation command, as per eStatusCode Values.
State	**Byte**	Current label activation state, as per eActivation Values.

5.7 Set Label Font Colour	Dec	254 21	LabelID R G B	2.4
	Hex	FE 15	LabelID R G B	
	ASCII	þ	LabelID R G B	

Set the font colour of an existing label. This command overrides the initial font colour, and immediately redraws the current text of the label in the new colour.

LabelID	Byte	Index used to identify this label in the label list.
R	Byte	Intensity of red, 0 to 255, limited to display metrics.
G	Byte	Intensity of green, 0 to 255, limited to display metrics.
B	Byte	Intensity of blue, 0 to 255, limited to display metrics.
Return Message	**252 21 Length**	**Result**
Result	Byte	Outcome of Set Label Font Colour command, as per eStatusCode Values.

5.8 Get Label Font Colour	Dec	254 22	LabelID	2.4
	Hex	FE 16	LabelID	
	ASCII	þ	LabelID	

Get the current font colour of an existing label.

LabelID	Byte	Index used to identify this label in the label list.
Return Message	**252 22 Length**	**Result R G B**
Result	Byte	Outcome of Get Label Font Colour command, as per eStatusCode Values.
R	Byte	Intensity of red, 0 to 255, limited to display metrics.
G	Byte	Intensity of green, 0 to 255, limited to display metrics.
B	Byte	Intensity of blue, 0 to 255, limited to display metrics.

5.9 Set Label Font Size	Dec	254 23	LabelID Size	2.4
	Hex	FE 17	LabelID Size	
	ASCII	þ	LabelID Size	

Set the font size of an existing label. This command overrides the initial font size, and immediately redraws the current text of the label in the new size.

LabelID	Byte	Index used to identify this label in the label list.
Size	Byte	New label size.
Return Message	**252 23 Length**	**Result**
Result	Byte	Outcome of Set Label Font Size command, as per eStatusCode Values.

5.10 Get Label Font Size	Dec	254 24	LabelID	2.4
	Hex	FE 18	LabelID	
	ASCII	þ	LabelID	

Get the current font size of an existing label.

LabelID	Byte	Index used to identify this label in the label list.
Return Message	**252 24 Length**	**Result Size**
Result	Byte	Outcome of Set Label Font Size command, as per eStatusCode Values.
Size	Byte	Current label size.

5.11 Print Unicode String	Dec	254 36	Text	2.0
	Hex	FE 24	Text	
	ASCII	þ $	Text	
Print a unicode formatted string to the current text window.				
Text	**Unicode String**	Unicode formatted string.		

5.12 Print UTF-8 String	Dec	254 37	Text	2.0
	Hex	FE 25	Text	
	ASCII	þ %	Text	
Print a UTF-8 formatted string to the current text window.				
Text	**UTF8 String**	UTF-8 formatted string.		

5.13 Set Control Character Mode	Dec	254 38	Mode	2.0
	Hex	FE 26	Mode	
	ASCII	þ &	Mode	
Set the behavior of the characters defined in the control characters section. Default is Unix mode.				
Mode	**Byte**	Desired control character mode, as per eControlCharacterMode Values.		

Table 22: eControlCharacterMode Values

Value	Description
0	Unix
1	Windows

5.14 Get Control Character Mode	Dec	254 39	2.0
	Hex	FE 27	
	ASCII	þ '	
Get the current control character mode.			
Return Message	**252 39 Length**	Mode	
Mode	**Byte**	Current control character mode, as per eControlCharacterMode Values.	

5.15 Get String Extents	Dec	254 42	Text	2.0
	Hex	FE 2A	Text	
	ASCII	þ *	Text	
Get the width and height of a box that a specific string would occupy if it was rendered on the GTT, with the current font. This command is useful for positioning and clearing text on the display.				
Text	**ASCII String**	String whose extents are desired.		
Return Message	**252 42 Length**	Width Height		
Width	**Short**	Width of the rendered string.		
Height	**Short**	Height of the rendered string.		

5.16 Set Text Window	Dec	254 43	X Y Width Height	2.0
	Hex	FE 2B	X Y Width Height	
	ASCII	þ +	X Y Width Height	

Set the position and size of the current text window on the screen. All future text insertion and print string commands will be confined to this window. The default window is the entire screen.

X	**Signed Short**	Leftmost coordinate of the text window.
Y	**Signed Short**	Topmost coordinate of the text window.
Width	**Short**	Width of the text window.
Height	**Short**	Height of the text window.

5.17 Get Text Window	Dec	254 44	2.0
	Hex	FE 2C	
	ASCII	þ ,	

Get the position and size of the current text window.

Return Message	**252 44 Length**	**X Y Width Height**
X	**Signed Short**	Leftmost coordinate of the text window.
Y	**Signed Short**	Topmost coordinate of the text window.
Width	**Short**	Height of the text window.
Height	**Short**	Height of the text window.

5.18 Reset Font	Dec	254 45	2.0
	Hex	FE 2D	
	ASCII	þ -	

Reset the font at ID 0 to the default GTT proggy style, with the last selected text colour.

5.19 Set Text Colour	Dec	254 46	R G B	2.0
	Hex	FE 2E	R G B	
	ASCII	þ .	R G B	

Set the colour of the current font used for all print string and create label commands sent to the GTT. Existing text and other fonts are not affected. The default text colour is white.

R	**Byte**	Intensity of red, 0 to 255, limited to display metrics.
G	**Byte**	Intensity of green, 0 to 255, limited to display metrics.
B	**Byte**	Intensity of blue, 0 to 255, limited to display metrics.

5.20 Get Text Colour	Dec	254 47	2.0
	Hex	FE 2F	
	ASCII	þ /	

Get the colour of the current font used to render all print string and create label commands.

Return Message	**252 47 Length**	**R G B**
R	**Byte**	Intensity of red, 0 to 255, limited to display metrics.
G	**Byte**	Intensity of green, 0 to 255, limited to display metrics.
B	**Byte**	Intensity of blue, 0 to 255, limited to display metrics.

5.21 Get Font	Dec	254 48		2.0
	Hex	FE 30		
	ASCII	þ 0		
Get the current font index used to render all print string and create label commands.				
Return Message	**252 48 Length**	**FontID**		
FontID	**Byte**	Font index used to identify the current font file in the font buffer.		

5.22 Set Font	Dec	254 49	FontID	2.0
	Hex	FE 31	FontID	
	ASCII	þ 1	FontID	
Set the font index that is used to render all print string and create label commands sent to the GTT. The default font index is 0, which is loaded with the proggy font on startup.				
FontID	**Byte**	Font index used to identify the desired font file in the font buffer.		
Return Message	**252 49 Length**	**Result**		
Result	**Byte**	Outcome of Set Current Font command, as per eStatusCode Values.		

5.23 Set Font Size	Dec	254 51	PtSize	2.0
	Hex	FE 33	PtSize	
	ASCII	þ 3	PtSize	
Set the size of the current font used to render all print string and create label commands sent to the GTT. The default font size is 24 point. Note that the proggy font has one size only.				
PtSize	**Byte**	Desired point size for the current font.		

5.24 Get Font Size	Dec	254 61		2.1
	Hex	FE 3D		
	ASCII	þ =		
Get the size of the current font used to render all print string and create label commands.				
Return Message	**252 61 Length**	**PtSize**		
PtSize	**Byte**	Implemented point size for the current font.		

5.25 Go Home	Dec	254 72		2.0
	Hex	FE 48		
	ASCII	þ H		
Set the text insertion point to the upper leftmost corner of the current text window.				

5.26 Set Text Insertion Point	Dec	254 121	X Y	2.0
	Hex	FE 79	X Y	
	ASCII	þ y	X Y	
Set the upper left coordinate of the next printed string to be displayed, relative to the current text window.				
X	**Signed Short**	Desired leftmost coordinate of the insertion point.		
Y	**Signed Short**	Desired topmost coordinate of the insertion point.		

5.27 Get Text Insertion Point	Dec	254 122		2.0
	Hex	FE 7A		
	ASCII	þ z		

Get the upper left coordinate of the next printed string to be displayed within the current text window.

Return Message	**252 122 Length**	X Y
X	**Signed Short**	Current leftmost coordinate of the insertion point.
Y	**Signed Short**	Current topmost coordinate of the insertion point.

5.28 Set Font Rendering Style	Dec	254 211	RenderType	2.0
	Hex	FE D3	RenderType	
	ASCII	þ Ó	RenderType	

Set the rendering style of the current font used for all print string and create label commands. Greyscale offers a more polished appearance at the cost of performance. Default is greyscale.

RenderType	**Byte**	Rendertype, as per eFontRenderType Values.

Table 23: eFontRenderType Values

Value	Description
0	Grayscale
1	Monochrome

5.29 Set Font Anchor Style	Dec	254 212	AnchorType	2.0
	Hex	FE D4	AnchorType	
	ASCII	þ Ô	AnchorType	

Set the anchoring style of the current text window font. Note that labels use only BaseLine rendering. The default style for text windows is UpperLeft.

AnchorType	**Byte**	Type of anchor, as per eAnchor.

Table 24: eAnchorType Values

Value	Description
0	UpperLeft
1	BaseLine

2.6 Bitmaps

6.1 Display Bitmap	Dec	254 97	BitmapID X Y	2.0
	Hex	FE 61	BitmapID X Y	
	ASCII	þ a	BitmapID X Y	

Display a bitmap image on the screen, from the bitmap buffer.

BitmapID	**Byte**	Index used to identify the desired file in the bitmap buffer.
X	**Signed Short**	Leftmost coordinate.
Y	**Signed Short**	Topmost coordinate.
Return Message	**252 97 Length**	Result
Result	**Byte**	Outcome of Display Bitmap command, as per eStatusCode Values.

6.2 Set Bitmap Transparency	Dec	254 98	BitmapID R G B	2.0
	Hex	FE 62	BitmapID R G B	
	ASCII	þ b	BitmapID R G B	

Set the transparent colour for all future renderings of a specific bitmap index. Does not affect previously drawn versions of the specified bitmap.

BitmapID	**Byte**	Index used to identify the desired file in the bitmap buffer.
R	**Byte**	Intensity of red, 0 to 255, limited to display metrics.
G	**Byte**	Intensity of green, 0 to 255, limited to display metrics.
B	**Byte**	Intensity of blue, 0 to 255, limited to display metrics.
Return Message	**252 98 Length**	Result
Result	**Byte**	Outcome of Set Bitmap Transparency command, as per eStatusCode Values.

2.7 NineSlices

7.1 Display 9-Slice	Dec	254 145	NineSliceID X Y Width Height	2.0
	Hex	FE 91	NineSliceID X Y Width Height	
	ASCII	þ ‘	NineSliceID X Y Width Height	

Display a 9-slice image on the screen, from the 9-slice buffer.

NineSliceID	**Byte**	Index used to identify the desired file in the 9-slice buffer.
X	**Signed Short**	Leftmost coordinate.
Y	**Signed Short**	Topmost coordinate.
Width	**Short**	Width of the 9-slice.
Height	**Short**	Height of the 9-slice.

2.8 Animations

8.1 Set Up Animation	Dec	254 193	AnimationID AnimationInstance X Y	2.0
	Hex	FE C1	AnimationID AnimationInstance X Y	
	ASCII	þ Á	AnimationID AnimationInstance X Y	

Define a region of the screen to be used for the specified animation. If an animation is already in use at that index, it will be overwritten. Multiple Animation Instances can be setup from one buffered animation file. Use the start animation command to display and play an animation instance.

AnimationID	**Byte**	Index where an animation file has been loaded.
AnimationInstance	**Byte**	Index used to identify this animation instance in the animation list.
X	**Signed Short**	Leftmost coordinate.
Y	**Signed Short**	Topmost coordinate.

8.2 Start/Stop Animation	Dec	254 194	AnimationInstance State	2.0
	Hex	FE C2	AnimationInstance State	
	ASCII	þ Â	AnimationInstance State	
Start or stop an animation instance. After it is started, an animation will loop until stopped.				
AnimationInstance	**Byte**	Index used to identify this animation instance in the animation list.		
State	**Byte**	Desired animation state, as per eAnimationState Values.		

Table 25: eAnimationState Values

Value	Description
0	Paused
1	Playing

8.3 Set Animation Frame	Dec	254 195	AnimationInstance Frame	2.0
	Hex	FE C3	AnimationInstance Frame	
	ASCII	þ Ã	AnimationInstance Frame	
Set the current frame of a displayed animation. If the frame exceeds the total number present, the animation will be set to the first frame.				
AnimationInstance	**Byte**	Index used to identify this animation instance in the animation list.		
Frame	**Byte**	Number of the frame to be displayed.		

8.4 Get Animation Frame	Dec	254 196	AnimationInstance	2.0
	Hex	FE C4	AnimationInstance	
	ASCII	þ Ä	AnimationInstance	
Get the current frame of an existing animation instance.				
AnimationInstance	**Byte**	Index used to identify this animation instance in the animation list.		
Return Message	**252 196 Length**	**Frame**		
Frame	**Byte**	Current state of the specified animation frame; 0 for paused, 1 for playing, 6 for invalid index.		

8.5 Stop All Animations	Dec	254 198	2.0
	Hex	FE C6	
	ASCII	þ Æ	
Stop all currently running animation instances at their present frame.			

8.6 Clear Animation	Dec	254 199	AnimationInstance	2.0
	Hex	FE C7	AnimationInstance	
	ASCII	þ Ç	AnimationInstance	

Stop the specified animation instance at the current frame and remove it from the animation list. The animation image data will remain loaded in the animation buffer and can be reused by issuing the setup command.

AnimationInstance	**Byte**	Index used to identify this animation instance in the animation list.

8.7 Clear All Animations	Dec	254 200	2.0
	Hex	FE C8	
	ASCII	þ È	

Stop all animation instances at their current frames and remove them from the animation list. The animation image data will remain loaded in animation buffers and can be reused by issuing the setup command.

8.8 Resume All Animations	Dec	254 201	2.0
	Hex	FE C9	
	ASCII	þ É	

Resume all stopped animation instances from their present frame.

2.9 Graphs

9.1 List All Bargraphs	Dec	254 102	2.0
	Hex	FE 66	
	ASCII	þ f	

Get the current state, type, and value of all 256 bargraphs in the bargraph list. Three bytes per entry indicate current display use, type, and current value.

Return Message	**252 102 Length**	**BarType BarValue**
BarType	**Byte**	Type of bargraph entry.
BarValue	**Signed Short**	Current value of bargraph entry.

Table 26: eBargraphType Values

Value	Description
0	Unused
1	Plain
2	NineSlice

9.2 Define a Plain Bargraph	Dec	254 103	BarID Min Max X Y Width Height FGR FGG FGB BGR BGG BGB D	2.0
	Hex	FE 67	BarID Min Max X Y Width Height FGR FGG FGB BGR BGG BGB D	
	ASCII	þ g	BarID Min Max X Y Width Height FGR FGG FGB BGR BGG BGB D	

Define a new plain bargraph. Upon execution of an update command, the bargraph are will be filled with the background colour, then a bar will be drawn to the current value using the foreground colour. New index definitions will overwrite old, invalid directions will default to 0, and inverted min and max values will be automatically corrected.

BarID	Byte	Index used to identify this bargraph in the bargraph list.
Min	Signed Short	Minimum bargraph value.
Max	Signed Short	Maximum bargraph value.
X	Signed Short	Leftmost coordinate of the bargraph.
Y	Signed Short	Topmost coordinate of the bargraph.
Width	Signed Short	Width of the bargraph.
Height	Signed Short	Height of the bargraph.
FGR	Byte	Red component of the foreground colour.
FGG	Byte	Green component of the foreground colour.
FGB	Byte	Blue component of the foreground colour.
BGR	Byte	Red component of the background colour.
BGG	Byte	Green component of the background colour.
BGB	Byte	Blue component of the background colour.
D	Byte	Direction that the bargraph will take, as per eBargraphOrientation Values.

Table 27: eBargraphOrientation Values

Value	Description
0	BottomToTop
1	LeftToRight
2	RightToLeft
3	TopToBottom

9.3 Define a 9-Slice Bargraph	Dec	254 104	BarID Min Max X Y Width Height BFG BBG D	2.0
	Hex	FE 68	BarID Min Max X Y Width Height BFG BBG D	
	ASCII	þ h	BarID Min Max X Y Width Height BFG BBG D	

Define a new 9-slice bargraph. Upon execution of an update command, the bargraph region will be filled with the background 9-slice, then a bar will be drawn to the current value using the foreground 9-slice. New index definitions will overwrite old, invalid directions will default to 0, and inverted min and max values will be automatically corrected.

BarID	Byte	Index used to identify this bargraph in the bargraph list.
Min	Signed Short	Minimum bargraph value.
Max	Signed Short	Maximum bargraph value.
X	Signed Short	Leftmost coordinate of the bargraph.
Y	Signed Short	Topmost coordinate of the bargraph.
Width	Signed Short	Width of the bargraph.
Height	Signed Short	Height of the bargraph.
BFG	Byte	9-Slice buffer index of the foreground image.
BBG	Byte	9-Slice buffer index of the background image.
D	Byte	Direction that the bargraph will take, as per eBargraphOrientation Values.

9.4 Update a Bargraph Value	Dec	254 105	BarID Value	2.0
	Hex	FE 69	BarID Value	
	ASCII	þ i	BarID Value	

Update the value of an existing bargraph. Value will be bounded to the bargraph minimum and maximum.

BarID	**Byte**	Index used to identify this bargraph in the bargraph list.
Value	**Signed Short**	Current value of the bargraph.
Return Message	**252 105 Length**	Result
Result	**Byte**	Outcome of Update a Bargraph Value command, as per eStatusCode Values.

9.5 Update Multiple Bargraph Values	Dec	254 106	BarID Length Values	2.0
	Hex	FE 6A	BarID Length Values	
	ASCII	þ j	BarID Length Values	

Update the values of multiple existing bargraphs. Values will be bounded to each bargraph minimum and maximum.

BarID	**Byte**	Index used to identify the first bargraph to be updated in the bargraph list.
Length	**Byte**	Length of the data to be transferred, in bytes.
Values	**Signed Short(s)**	Current values, one for each bargraph index to be updated.
Return Message	**252 106 Length**	Result
Result	**Byte**	Outcome of Set Multiple Bargraph Values command, as per eStatusCode Values.

9.6 Clear All Bargraphs	Dec	254 107	2.0
	Hex	FE 6B	
	ASCII	þ k	

Clear all data from the bargraph list. This command erases all attributes and sets all bargraphs to an unused state, but does not affect the screen visually.

9.7 Reset a Trace Value	Dec	254 109	TraceID	2.1
	Hex	FE 6D	TraceID	
	ASCII	þ m	TraceID	

Clear all visual data from a trace, and reset its value. As a result, the next Update Trace command behaves as though it is the very first update after initialization.

TraceID	**Byte**	Index used to identify this trace in the trace list.

9.8 Reset Multiple Trace Values	Dec	254 110	TraceID Number	2.1
	Hex	FE 6E	TraceID Number	
	ASCII	þ n	TraceID Number	

Clear all visual data from multiple traces, and reset their values. As a result, the next Update Trace commands behave as though they are the very first updates after initialization.

TraceID	**Byte**	Index used to identify the first trace to be reset in the trace list.
Number	**Byte**	Number of trace entries to be reset.

9.9 List All Traces	Dec	254 115		2.1
	Hex	FE 73		
	ASCII	þ s		
Get the current state and value of all 256 traces in the trace list.				
Return Message	**252 115 Length**	**TraceID Value**		
TraceID	**Byte**	Trace index number. One for each entry. 0 signifies an undefined entry.		
Value	**Signed Short**	Current value of the trace. One for each entry.		

9.10 Initialize a Trace	Dec	254 116	TraceID X Y Width Height Min Max Step Style Red Green Blue	2.1
	Hex	FE 74	TraceID X Y Width Height Min Max Step Style Red Green Blue	
	ASCII	þ t	TraceID X Y Width Height Min Max Step Style Red Green Blue	
Initialize a new graph trace. Upon execution of an update command, the trace region will be shifted by the step size, and a line or bar drawn between the previous value and the new one. A multi-trace graph can be created by initializing traces with the same area, step, and style. Multi-trace graphs can be updated with the Update Multiple Traces command, individual traces can be updated with the Update a Trace command.				
TraceID	**Byte**	Index used to identify this trace in the trace list.		
X	**Signed Short**	Leftmost coordinate of the trace region.		
Y	**Signed Short**	Topmost coordinate of the trace region.		
Width	**Signed Short**	Width of the trace region.		
Height	**Signed Short**	Height of the trace region.		
Min	**Signed Short**	Value displayed at the lowest point of the trace.		
Max	**Signed Short**	Value displayed at the highest point of the trace.		
Step	**Byte**	Number of pixels shifted when a trace is updated.		
Style	**Byte**	Orientation and Direction of the trace, as per eTraceTypeandDirection Values. A style is created by summing values of individual attributes. For example, a Line with a Bottom Left origin, Shifting right has a Style value of 129.		
Red	**Byte**	Intensity of red for trace colour, 0 to 255, limited to display metrics.		
Green	**Byte**	Intensity of green for trace colour, 0 to 255, limited to display metrics.		
Blue	**Byte**	Intensity of blue for trace colour, 0 to 255, limited to display metrics.		

Table 28: eTraceTypeandDirection Values

Value	Description
0	Bar
1	Line
2	Step
3	Box
0	BottomLeft
0	ShiftTowardOrigin
16	LeftUp
32	TopRight
48	RightDown
64	BottomRight
128	ShiftAwayFromOrigin
80	LeftDown
96	TopLeft
112	RightUp

9.11 Update a Trace	Dec	254 117	TraceID Value	2.1
	Hex	FE 75	TraceID Value	
	ASCII	þ u	TraceID Value	

Update the value of the trace at the specified index. Trace will be bounded to the minimum and maximum.

TraceID	Byte	Index used to identify this trace in the trace list.
Value	Signed Short	Current value of the specified trace.

9.12 Update Multiple Traces	Dec	254 118	TraceID Length Values	2.1
	Hex	FE 76	TraceID Length Values	
	ASCII	þ v	TraceID Length Values	

Simultaneously update the values of the specified traces, useful for updating a multi-variable graph.

TraceID	Byte	Index used to identify the first trace to be updated in the trace list.
Length	Byte	Length of the data to be transferred, in bytes.
Values	Signed Short(s)	Current values, one for each of the trace index to be updated.
Return Message	**252 118 Length**	Result
Result	Byte	Outcome of Update Multiple Traces command, as per eStatusCode Values.

9.13 Clear All Traces	Dec	254 119	2.1
	Hex	FE 77	
	ASCII	þ w	

Clear all data from the trace list. This command erases all attributes and sets all traces to an unused state, but does not affect the screen visually.

9.14 Set Trace Min and Max Values	Dec	254 148	TraceID Min Max	2.2
	Hex	FE 94	TraceID Min Max	
	ASCII	þ ”	TraceID Min Max	

Update the min and max values of the specified trace. Trace will visually update to new bounds.

TraceID	Byte	Index used to identify the previously defined trace. Specific to Traces.
Min	Signed Short	The new minimum value for the trace as specified by TraceIndex.
Max	Signed Short	The new maximum value for the trace as specified by TraceIndex.

9.15 Get Trace Min and Max Values	Dec	254 149	TraceID	2.2
	Hex	FE 95	TraceID	
	ASCII	þ •	TraceID	

Get the current min and max values of the specified trace.

TraceID	Byte	Index used to identify the previously defined trace. Specific to Traces.
Return Message	**252 149 Length**	Min Max
Min	Signed Short	The min value of the trace specified.
Max	Signed Short	The max value of the trace specified.

2.10 Keypad

10.1 Clear Key Buffer	Dec	254 69	2.0
	Hex	FE 45	
	ASCII	þ E	

Clear all saved key presses from the key buffer.

10.2 Set Keypad Transmit Mode	Dec	254 79	AutoTransmit	2.0
	Hex	FE 4F	AutoTransmit	
	ASCII	þ O	AutoTransmit	

Toggle auto transmission of key values. Can be used to poll the key buffer.

AutoTransmit	Byte	Auto transmit mode, as per eOnOff Values.

Table 29: eOnOff Values

Value	Description
0	Off
1	On

10.3 Set Debounce Time	Dec	254 85	Mode	2.0
	Hex	FE 55	Mode	
	ASCII	þ U	Mode	

Set the time, in ms, between a key press and a key read by the display. Most switches will bounce when pressed; the debounce time allows the switch to settle for an accurate read. Default is 64ms.

Mode	Byte	Debounce time in milliseconds.

10.4 Set Typematic Interval	Dec	254 158	Interval	2.0
	Hex	FE 9E	Interval	
	ASCII	þ ž	Interval	

Set the interval between reported key presses when a key is held and the display is in typematic mode.

Interval	Short	Time between key reports, in ms, default is 200ms.

10.5 Set Typematic Delay	Dec	254 159	Delay	2.0
	Hex	FE 9F	Delay	
	ASCII	þ Ÿ	Delay	

Set the delay between the first key press and first typematic report when a key is held in typematic mode.

Delay	Short	Time key must be held to trigger typematic reports, in ms, default is 1000ms.

10.6 Set Auto Repeat Mode	Dec	254 165	Mode	2.0
	Hex	FE A5	Mode	
	ASCII	þ ¥	Mode	

Set key press repeat mode to typematic or hold. In typematic mode if a key press is held, by default the key value is transmitted immediately, then 5 times a second after a 1 second delay. In hold mode, the key down value is transmitted once when pressed, and then the key up value is sent when the key is released. Default is typematic.

Mode	Byte	Desired keypad auto repeat mode, as per eKeypadRepeatMode Values.

Table 30: eKeypadRepeatMode Values

Value	Description
0	Off
1	Hold
2	Typematic

10.7 Assign Keypad Codes	Dec	254 213	Length KeyCodes	
	Hex	FE D5	Length KeyCodes	
	ASCII	þ Õ	Length KeyCodes	
Assign the values sent to the host when a key press is detected. Up to 25 keys may be defined.				
Length	**Byte**	Length of the data to be transferred, in bytes.		
KeyCodes	**Byte(s)**	A list of byte values for each key to be defined. Default values are 65 through 90.		

2.11 Touch

11.1 Create a Touch Region	Dec	254 132	RegionID X Y Width Height Up Down	2.0
	Hex	FE 84	RegionID X Y Width Height Up Down	
	ASCII	þ „	RegionID X Y Width Height Up Down	
Create a region of the screen that responds to touch events with a unique message and momentary visual update.				
RegionID	**Byte**	Index used to identify this touch region in the touch region list. Region 255 is reserved for out of region responses.		
X	**Signed Short**	Leftmost coordinate of the touch region.		
Y	**Signed Short**	Topmost coordinate of the touch region.		
Width	**Short**	Width of the touch region.		
Height	**Short**	Height of the touch region.		
Up	**Byte**	Index of the loaded bitmap displayed when the region is released.		
Down	**Byte**	Index of the loaded bitmap displayed when the region is touched.		

11.2 Clear a Touch Region	Dec	254 133	RegionID	2.0
	Hex	FE 85	RegionID	
	ASCII	þ …	RegionID	
Clear the specified touch region from the touch region list. This ensures touch events will no longer be reported from this region.				
RegionID	**Byte**	Index used to identify this touch region in the touch region list.		

11.3 Clear All Touch Regions	Dec	254 134		2.0
	Hex	FE 86		
	ASCII	þ †		
Clear all touch regions from the screen and memory, ensuring their touch events will no longer be reported.				

11.4 Change Touch Reporting Style	Dec	254 135	ReportingType	2.0
	Hex	FE 87	ReportingType	
	ASCII	þ ‡	ReportingType	
Customize the way in which touch events are reported. Default is RegionDown.				
ReportingType	Byte	Desired touch reporting style, as per eTouchReportingType Values.		

Table 31: eTouchReportingType Values

Value	Description
0	RegionNone
1	RegionDown
2	RegionUp
3	RegionUpDown
4	RegionMove
5	RegionMoveDown
6	RegionMoveUp
7	RegionMoveUpDown
8	CoordNone
9	CoordDown
10	CoordUp
11	CoordUpDown
12	CoordMove
13	CoordMoveDown
14	CoordMoveUp
15	CoordMoveUpDown

11.5 Get Touch Reporting Style	Dec	254 136		2.0
	Hex	FE 88		
	ASCII	þ ˆ		
Get the current touch reporting style.				
Return Message	252 136 Length	Result ReportingType		
Result	Byte	Outcome of Get Touch Reporting Style command, as per eStatusCode Values.		
ReportingType	Byte	Current touch reporting style, as per eTouchReportingType Values.		

11.6 Set Dragging Threshold	Dec	254 137	Threshold	2.0
	Hex	FE 89	Threshold	
	ASCII	þ ‰	Threshold	
Set the distance a press is required to travel before a move event is reported. Precision will vary inversely to data transmitted; care should be taken to find a suitable balance. Distance is calculated as [Δx]^2 + [Δy]^2 = d^2.				
Threshold	Short	Dragging threshold value. Default is 3 pixels.		

11.7 Calibrate Touch Screen	Dec	254 139	2.0
	Hex	FE 8B	
	ASCII	þ ‹	

Initiate the touch screen calibration sequence, after user input is complete a confirmation byte will be returned, new calibration settings will be loaded, and the calibration will be saved as \SYSTEM\touchcal.dat. Calibration can be restored from the file at any time.

Return Message	252 139 Length	Result
Result	Byte	Outcome of Calibrate Touch Screen command, as per eCalibrationErrorCode Values.

Table 32: eCalibrationErrorCode Values

Value	Description
1	CalibrationSuccessful
12	CalibrationInvalid

11.8 Load Region File	Dec	254 140	FileName	2.0
	Hex	FE 8C	FileName	
	ASCII	þ Œ	FileName	

Load a group of touch region definitions from a file. If an existing region exists with the same index as a region in the file, it will be overwritten. See the Region File example.

FileName	ASCII String	Filename, and path from the root folder, of the region file to load.
Return Message	252 140 Length	Result
Result	Byte	Outcome of Load Region File command, as per eStatusCode Values.

11.9 Restore Touch Calibration	Dec	254 141	2.0
	Hex	FE 8D	
	ASCII	þ •	

Restore touch calibration using the data from \SYSTEM\touchcal.dat•, if this file is present.

Return Message	252 141 Length	Result
Result	Byte	Outcome of Restore Touch Calibration command, as per eRestoreCalibrationErrorCode Values.

Table 33: eRestoreCalibrationErrorCode Values

Value	Description
0	RestoreCalibrationInvalid
1	RestoreCalibrationSuccessful

11.10 Set Out of Region Setting	Dec	254 142	Setting	2.0
	Hex	FE 8E	Setting	
	ASCII	þ Ž	Setting	

Set whether out of region responses will be returned or not. Out of region responses are returned when a touch occurs outside a region, while in region mode. The index of an out of region response is 255.

Setting	Byte	Desired out of region setting, as per eOnOff Values. Default is Off.

11.11 Get Out of Region Setting	Dec Hex ASCII	254 143 FE 8F þ •		2.0
Get the current out of region setting.				
Return Message	**252 143 Length**	Report		
Report	**Byte**	Current out of region setting, as per eOnOff Values.		

11.12 Set Region Activate State	Dec Hex ASCII	254 146 FE 92 þ ’	RegionID Enable RegionID Enable RegionID Enable	2.2
Set the activation state for a specific region. Useful for temporarily disabling a region. When a region is created, the default is activated.				
RegionID	**Byte**	Index used to identify the touch region in the touch region list.		
Enable	**Byte**	Activation state, as per eEnable Values.		
Return Message	**252 146 Length**	Result		
Result	**Byte**	Outcome of Set Region Activation State command, as per eStatusCode Values.		

11.13 Get Region Activate State	Dec Hex ASCII	254 147 FE 93 þ “	RegionID RegionID RegionID	2.2
Get the current activation state of a specific region. An invalid touch region error will be returned if the specified index does not exist in the touch region list.				
RegionID	**Byte**	Index used to identify the touch region in the touch region list.		
Return Message	**252 147 Length**	Enable		
Enable	**Byte**	Current region activation state, as per eEnable Values. Invalid indices return 16.		

11.14 Create a Toggle Region	Dec Hex ASCII	254 150 FE 96 þ –	RegionID X Y Width Height OffID OnID RegionID X Y Width Height OffID OnID RegionID X Y Width Height OffID OnID	2.4
Create a region of the screen that responds to touch events with a unique message and toggleable visual update.				
RegionID	**Byte**	Index used to identify this toggle region in the touch region list. Region 255 is reserved for out of region responses.		
X	**Signed Short**	Leftmost coordinate of the toggle region.		
Y	**Signed Short**	Topmost coordinate of the toggle region.		
Width	**Short**	Width of the toggle region.		
Height	**Short**	Height of the toggle region.		
OffID	**Byte**	Index of the loaded bitmap displayed when the region is in an inactive state.		
OnID	**Byte**	Index of the loaded bitmap displayed when the region is in a toggled state.		
Return Message	**252 150 Length**	Result		
Result	**Byte**	Outcome of the Create Toggle Region command, as per eStatusCode Values.		

11.15 Create a Slider				
	Dec	254 161	RegionID X Y LT RB TrkWidth TrkHeight BtnWidth BtnHeight TrkID BtnID Style	2.4
	Hex	FE A1	RegionID X Y LT RB TrkWidth TrkHeight BtnWidth BtnHeight TrkID BtnID Style	
	ASCII	þ ¡	RegionID X Y LT RB TrkWidth TrkHeight BtnWidth BtnHeight TrkID BtnID Style	

Create a region of the screen that displays a slider control and responds to touch events with a unique message including its current value, as well as a matching visual update.

RegionID	**Byte**	Index used to identify this slider in the touch region list. Region 255 is reserved for out of region responses.
X	**Signed Short**	Leftmost coordinate of the slider region.
Y	**Signed Short**	Topmost coordinate of the slider region.
LT	**Signed Short**	Leftmost/Topmost value returned by the slider region. Default initial button location.
RB	**Signed Short**	Rightmost/Bottommost value returned by the slider region.
TrkWidth	**Short**	Width of the slider track region.
TrkHeight	**Short**	Height of the slider track region.
BtnWidth	**Short**	Width of the slider button region.
BtnHeight	**Short**	Height of the slider button region.
TrkID	**Byte**	Index of the loaded 9-slice file displayed within the track region.
BtnID	**Byte**	Index of the loaded 9-slice file displayed within the button region.
Style	**Byte**	Style of the slider, as per eSliderStyles Values.
Return Message	**252 161 Length**	**Result**
Result	**Byte**	Outcome of the Create Slider command, as per eStatusCode Values.

Table 34: eSliderStyles Values

Value	Description
0	Vertical
1	Horizontal

11.16 Create a Filled Slider	Dec	254 163	RegionID X Y LT RB TrkWidth TrkHeight BtnWidth BtnHeight TrkID FilID BtnID Style	2.6
	Hex	FE A3	RegionID X Y LT RB TrkWidth TrkHeight BtnWidth BtnHeight TrkID FilID BtnID Style	
	ASCII	þ £	RegionID X Y LT RB TrkWidth TrkHeight BtnWidth BtnHeight TrkID FilID BtnID Style	

Create a region of the screen that displays a filled slider control and returns touch events.

RegionID	**Byte**	Index used to identify this filled slider in the touch region list. Region 255 is reserved for out of region responses.
X	**Signed Short**	Leftmost coordinate of the filled slider region.
Y	**Signed Short**	Topmost coordinate of the filled slider region.
LT	**Signed Short**	Leftmost/Topmost value returned by the filled slider region. Default initial button location.
RB	**Signed Short**	Rightmost/Bottommost value returned by the filled slider region.
TrkWidth	**Short**	Width of the slider track region.
TrkHeight	**Short**	Height of the slider track region.
BtnWidth	**Short**	Width of the slider button region.
BtnHeight	**Short**	Height of the slider button region.
TrkID	**Byte**	Index of the loaded 9-slice file displayed within the empty track region.
FilID	**Byte**	Index of the loaded 9-slice file displayed within the filled track region.
BtnID	**Byte**	Index of the loaded 9-slice file displayed within the button region.
Style	**Byte**	Style of the slider, as per eSliderStyles Values.
Return Message	**252 163 Length**	**Result**
Result	**Byte**	Outcome of the Create Slider command, as per eStatusCode Values.

11.17 Set Slider Value	Dec	254 166	RegionID Value	2.4
	Hex	FE A6	RegionID Value	
	ASCII	þ ¦	RegionID Value	

Set the value of a previously created slider. Useful for setting the initial slider position.

RegionID	**Byte**	Index used to identify the slider in the touch region list.
Value	**Signed Short**	Desired value for the specified slider.
Return Message	**252 166 Length**	**Result**
Result	**Byte**	Outcome of the Set Slider Value command, as per eStatusCode Values.

11.18 Get Slider Value	Dec	254 167	RegionID	2.4
	Hex	FE A7	RegionID	
	ASCII	þ §	RegionID	

Get the current value of an existing slider.

RegionID	**Byte**	Index used to identify the slider in the touch region list.
Return Message	**252 167 Length**	**Result Value**
Result	**Byte**	Outcome of the Get Slider Value command, as per eStatusCode Values.
Value	**Signed Short**	Current value of the specified slider.

11.19 Set Toggle State	Dec	254 170	RegionID State	2.5
	Hex	FE AA	RegionID State	
	ASCII	þ ª	RegionID State	

Set the state of a previously created toggle region. Used for setting the initial toggle position, or controlling a toggleable output object.

RegionID	Byte	Index used to identify the toggle region in the touch region list.
State	Byte	Desired state for the specified toggle region.
Return Message	252 170 Length	Result
Result	Byte	Outcome of the Set Toggle State command, as per eStatusCode Values.

11.20 Get Toggle State	Dec	254 171	RegionID	2.5
	Hex	FE AB	RegionID	
	ASCII	þ «	RegionID	

Get the state of a previously created toggle region.

RegionID	Byte	Index used to identify the toggle region in the touch region list.
Return Message	252 171 Length	Result State
Result	Byte	Outcome of the Get Toggle State command, as per eStatusCode Values.
State	Byte	Current state of the specified toggle region.

2.12 Output

12.1 Set GPO State	Dec	254 73	Number Setting	2.0
	Hex	FE 49	Number Setting	
	ASCII	þ I	Number Setting	

Toggle the specified General Purpose Output pin on or off, sourcing up to 15mA current at 5V per GPO or sinking to ground. This command can be used to control devices, or signal a host device.

Number	Byte	GPO to be controlled.
Setting	Byte	GPO state, as per eGPOSetting Values.

Table 35: eGPOSetting Values

Value	Description
1	On
0	Off

12.2 Activate Motor	Dec	254 160	Frequency Duration	2.0
	Hex	FE A0	Frequency Duration	
	ASCII	þ	Frequency Duration	

Activate a vibratory pulse from the motor at the specified frequency for the defined duration.

Frequency	Short	Frequency of the vibration in Hertz.
Duration	Short	Duration of the vibration in milliseconds.

12.3 Set Input Feedback	Dec	254 182	InputOutputType DownFrequency UpFrequency	2.0
	Hex	FE B6	InputOutputType DownFrequency UpFrequency	
	ASCII	þ ¶	InputOutputType DownFrequency UpFrequency	
Initiate autonomous feedback by specifying a 50ms output event for specific input events.				
InputOutputType	Byte	Desired input event and output response types, as per eKeypadInputOutputType Values. Multiple events and/or responses can be selected by summing values.		
DownFrequency	Short	Frequency of the down event in Hertz.		
UpFrequency	Short	Frequency of the up event in Hertz.		

Table 36: eKeypadInputOutputType Values

Value	Description
0	None
1	OutputBeep
2	OutputMotor
4	InputKeypad
8	InputTouch

12.4 Activate Buzzer and Motor	Dec	254 183	Frequency Duration	2.1
	Hex	FE B7	Frequency Duration	
	ASCII	þ ·	Frequency Duration	
Active both a vibratory pulse from the motor and a tone from the piezo buzzer simultaneously, at the specified frequency for the defined interval.				
Frequency	Short	Frequency of the beep and vibration in Hertz.		
Duration	Short	Duration of the beep in milliseconds.		

12.5 Activate Buzzer	Dec	254 187	Frequency Duration	2.0
	Hex	FE BB	Frequency Duration	
	ASCII	þ »	Frequency Duration	
Activate a tone from the piezo buzzer at the specified frequency for the defined duration.				
Frequency	Short	Frequency of the beep in Hertz.		
Duration	Short	Duration of the beep in milliseconds.		

12.6 Set Default Buzzer Beep	Dec	254 188	Frequency Duration	2.0
	Hex	FE BC	Frequency Duration	
	ASCII	þ ¼	Frequency Duration	
Set the frequency and duration of the default beep transmitted when the bell character is transmitted.				
Frequency	Short	Frequency of the beep in Hertz.		
Duration	Short	Duration of the beep in milliseconds.		

2.13 Scripts

13.1 Run Script File	Dec	254 93	FileName	2.0
	Hex	FE 5D	FileName	
	ASCII	þ]	FileName	

Run a script file from the GTT SD card. This command will process an array of bytes from a script file as if it was received from the serial port. Sending data to the serial port is still possible, but it will queue up in the input buffer and will only be parsed after the execution of the script file. Scripts may be stacked up to 10 deep.

FileName	**ASCII String**	Filename, and path from the root folder, of the script file to run.

13.2 Create a Scripted Region	Dec	254 131	RegionID X Y W H UpBitmap DownBitmap UpScript DownScript	2.1
	Hex	FE 83	RegionID X Y W H UpBitmap DownBitmap UpScript DownScript	
	ASCII	þ ƒ	RegionID X Y W H UpBitmap DownBitmap UpScript DownScript	

Create a region of the screen that responds to touch events with a unique message, momentary visual update, and script execution. Scripts will always execute, regardless of the current touch reporting style. If a script is not desired, use an empty string for its filename.

RegionID	**Byte**	Index used to identify this scripted region in the touch region list. Region 255 is reserved for out of region responses.
X	**Signed Short**	Leftmost coordinate of the scripted touch region.
Y	**Signed Short**	Topmost coordinate of the scripted touch region.
W	**Short**	Width of the scripted touch region.
H	**Short**	Height of the scripted touch region.
UpBitmap	**Byte**	Index of the loaded bitmap displayed when the region is released.
DownBitmap	**Byte**	Index of the loaded bitmap displayed when the region is pressed.
UpScript	**ASCII String**	Filename of the script to be executed when the region is released.
DownScript	**ASCII String**	Filename of the script to be executed when the region is pressed.

13.3 Create a Scripted Key	Dec	254 138	KeyID Row Col UpScript DownScript	2.1
	Hex	FE 8A	KeyID Row Col UpScript DownScript	
	ASCII	þ Š	KeyID Row Col UpScript DownScript	

Link the execution of a script file to a specific key value. Scripts always execute, regardless of the current key reporting style. If a script is not desired, use an empty string for its filename.

KeyID	**Byte**	Index used to identify the desired key value.
Row	**Byte**	Row index of the scripted key.
Col	**Byte**	Column index of the scripted key.
UpScript	**ASCII String**	Filename of the script to be executed when the key is released.
DownScript	**ASCII String**	Filename of the script to be executed when the key is pressed.

13.4 Create a Scripted Toggle Region	Dec	254 162	RegionID X Y Width Height OffID OnID OffScript OnScript	2.4
	Hex	FE A2	RegionID X Y Width Height OffID OnID OffScript OnScript	
	ASCII	þ ¢	RegionID X Y Width Height OffID OnID OffScript OnScript	

Create a region of the screen that responds to touch events with a unique message, toggleable visual update, and script execution. Scripts will always execute, regardless of the current touch reporting style. If a script is not desired, use an empty string for its filename.

RegionID	Byte	Index used to identify this scripted toggle region in the touch region list. Region 255 is reserved for out of region responses.
X	Signed Short	Leftmost coordinate of the scripted toggle region.
Y	Signed Short	Topmost coordinate of the scripted toggle region.
Width	Short	Width of the scripted toggle region.
Height	Short	Height of the scripted toggle region.
OffID	Byte	Index of the loaded bitmap displayed when the region is in an inactive state.
OnID	Byte	Index of the loaded bitmap displayed when the region is in a toggled state.
OffScript	ASCII String	Filename of the script to be executed when the region is first placed in an inactive state.
OnScript	ASCII String	Filename of the script to be executed when the region is first placed in a toggled state.
Return Message	252 162 Length	Result
Result	Byte	Outcome of the Create Scripted Toggle Region command, as per eStatusCode Values.

3 Appendix

3.1 Command Summary

Available commands below include identifying number, required parameters, the returned response and the response type.

Table 37: Communication Commands

Name	Dec	Hex	ASCII	Parameters	Response
Enter Mass Storage Mode	4	04	[EOT]	None	None
Set Communication Channel	5	05	[ENQ]	Channel	None
Set Baud Rate	57	39	9	BaudRate	None
Set Flow Control Mode	58	3A	:	FlowControl	None
Set I2C Address	247	F7	÷	I2Caddress	None
Echo	255	FF	ÿ	Message	ReturnMessage

Table 38: Module Commands

Name	Dec	Hex	ASCII	Parameters	Response
Get Protocol Revision	0	00	[NUL]	None	Major, Minor
Reset Module	1	01	[SOH]	None	None
Set Typematic Delay	2	02	[STX]	Time	None
Get Display Metrics	3	03	[ETX]	None	Width, Height, BitsRed, BitsGreen, BitsBlue
Set Screen Orientation	50	32	2	Orientation	None
Set Customer Data	52	34	4	Length, Data	None
Get Customer Data	53	35	5	None	Length, Data
Get Module Type	55	37	7	None	Module
Get Module String	56	38	8	None	ModuleString
Set Backlight Brightness	153	99	™	Brightness	None
Get Backlight Brightness	154	9A	š	None	Brightness
Write ScratchPad	204	CC	Ì	Index, Length, Data	None
Read ScratchPad	205	CD	Í	Index, Size	Length, Result

Table 39: Drawing Commands

Name	Dec	Hex	ASCII	Parameters	Response
Set Background Drawing Colour	86	56	V	R, G, B	None
Get Background Drawing Colour	87	57	W	None	R, G, B
Clear Screen	88	58	X	None	None
Scroll Screen	89	59	Y	X, Y, Width, Height, MoveX, MoveY	None
Enable Manual Update	90	5A	Z	Enable	None
Manual Update	91	5B	[	None	None
Flush Region	92	5C	\	X, Y, Width, Height	None
Set Drawing Colour	99	63	c	R, G, B	None
Get Drawing Colour	100	64	d	None	R, G, B
Continue Line	101	65	e	X, Y	None
Draw Line	108	6C	l	X1, Y1, X2, Y2	None
Draw Pixel	112	70	p	X, Y	None
Draw Rectangle	114	72	r	X, Y, Width, Height	None
Draw Filled Rectangle	120	78	x	X, Y, Width, Height	None
Draw Circle	123	7B	{	X, Y, Radius	None
Draw Filled Circle	124	7C	|	X, Y, Radius	None
Draw an Ellipse	125	7D	}	X, Y, XRadius, YRadius	None
Draw a Filled Ellipse	126	7E	~	X, Y, XRadius, YRadius	None
Draw Rounded Rectangle	127	7F	□	X, Y, Width, Height, Radius	None
Draw Filled Rounded Rectangle	128	80	€	X, Y, Width, Height, Radius	None
Draw Triangle	129	81	•	X1, Y1, X2, Y2, X3, Y3	None
Draw Filled Triangle	130	82	‚	X1, Y1, X2, Y2, X3, Y3	None

Table 40: Buffers Commands

Name	Dec	Hex	ASCII	Parameters	Response
Load Font	40	28	(	FontID, FileName	Result
Read Screen Rectangle	94	5E	^	X, Y, Width, Height, Format	Result, Format, Length, Data
Load Bitmap	95	5F	_	BitmapID, FileName	Result
Copy Screen Rectangle	96	60	`	BitmapID, X, Y, Width, Height	None
Load 9-Slice	144	90	•	NineSliceID, Filename	Result
Load Animation	192	C0	À	AnimationID, Filename	None
Clear a Buffer	208	D0	Ð	Type, ID	None
Clear All Buffers	209	D1	Ñ	None	None

Table 41: Text Commands

Name	Dec	Hex	ASCII	Parameters	Response
Create a Label	16	10	[DLE]	LabelID, X, Y, Width, Height, Rot, VJst, HJst, Font, R, G, B	None
Update a Label (UTF8)	17	11		LabelID, Format, Value	None
Update a Label (ASCII)	17	11		LabelID, Format, Value	None
Update a Label (Unicode)	17	11		LabelID, Format, Value	None
Set Label Activation State	19	13		LabelID, State	Result
Get Label Activation State	20	14		LabelID	Result, State
Set Label Font Colour	21	15		LabelID, R, G, B	Result
Get Label Font Colour	22	16		LabelID	Result, R, G, B
Set Label Font Size	23	17		LabelID, Size	Result
Get Label Font Size	24	18		LabelID	Result, Size
Print Unicode String	36	24	$	Text	None
Print UTF-8 String	37	25	%	Text	None
Set Control Character Mode	38	26	&	Mode	None
Get Control Character Mode	39	27	'	None	Mode
Get String Extents	42	2A	*	Text	Width, Height
Set Text Window	43	2B	+	X, Y, Width, Height	None
Get Text Window	44	2C	,	None	X, Y, Width, Height
Reset Font	45	2D	-	None	None
Set Text Colour	46	2E	.	R, G, B	None
Get Text Colour	47	2F	/	None	R, G, B
Get Font Size	48	30	0	None	FontID
Set Font Anchor Style	49	31	1	FontID	Result
Set Font Size	51	33	3	PtSize	None
Get Font Size	61	3D	=	None	PtSize
Go Home	72	48	H	None	None
Set Text Insertion Point	121	79	y	X, Y	None
Get Text Insertion Point	122	7A	z	None	X, Y
Set Font Rendering Style	211	D3	Ó	RenderType	None
Set Font Anchor Style	212	D4	Ô	AnchorType	None

Table 42: Bitmaps Commands

Name	Dec	Hex	ASCII	Parameters	Response
Display Bitmap	97	61	a	BitmapID, X, Y	Result
Set Bitmap Transparency	98	62	b	BitmapID, R, G, B	Result

Table 43: NineSlices Commands

Name	Dec	Hex	ASCII	Parameters	Response
Display 9-Slice	145	91	‘	NineSliceID, X, Y, Width, Height	None

Table 44: Animations Commands

Name	Dec	Hex	ASCII	Parameters	Response
Set Up Animation	193	C1	Á	AnimationID, AnimationInstance, X, Y	None
Start/Stop Animation	194	C2	Â	AnimationInstance, State	None
Set Animation Frame	195	C3	Ã	AnimationInstance, Frame	None
Get Animation Frame	196	C4	Ä	AnimationInstance	Frame
Stop All Animations	198	C6	Æ	None	None
Clear Animation	199	C7	Ç	AnimationInstance	None
Clear All Animations	200	C8	È	None	None
Resume All Animations	201	C9	É	None	None

Table 45: Graphs Commands

Name	Dec	Hex	ASCII	Parameters	Response
List All Bargraphs	102	66	f	None	BarType, BarValue
Define a Plain Bargraph	103	67	g	BarID, Min, Max, X, Y, Width, Height, FGR, FGG, FGB, BGR, BGG, BGB, D	None
Define a 9-Slice Bargraph	104	68	h	BarID, Min, Max, X, Y, Width, Height, BFG, BBG, D	None
Update a Bargraph Value	105	69	i	BarID, Value	Result
Update Multiple Bargraph Values	106	6A	j	BarID, Values	Result
Clear All Bargraphs	107	6B	k	None	None
Reset a Trace Value	109	6D	m	TraceID	None
Reset Multiple Trace Values	110	6E	n	TraceID, Number	None
List All Traces	115	73	s	None	TraceID, Value
Initialize a Trace	116	74	t	TraceID, X, Y, Width, Height, Min, Max, Step, Style, Red, Green, Blue	None
Update a Trace	117	75	u	TraceID, Value	None
Update Multiple Traces	118	76	v	TraceID, Values	Result
Clear All Traces	119	77	w	None	None
Set Trace Min and Max Values	148	94	”	TraceID, Min, Max	None
Get Trace Min and Max Values	149	95	•	TraceID	Min, Max

Table 46: Keypad Commands

Name	Dec	Hex	ASCII	Parameters	Response
Clear Key Buffer	69	45	E	None	None
Set Keypad Transmit Mode	79	4F	O	AutoTransmit	None
Set Debounce Time	85	55	U	Mode	None
Set Typematic Interval	158	9E	ž	Interval	None
Set Typematic Delay	159	9F	Ÿ	Delay	None
Set Auto Repeat Mode	165	A5	¥	Mode	None
Assign Keypad Codes	213	D5	Õ	Length, KeyCodes	None

Table 47: Touch Commands

Name	Dec	Hex	ASCII	Parameters	Response
Create a Touch Region	132	84	„	RegionID, X, Y, Width, Height, Up, Down	None
Clear a Touch Region	133	85	…	RegionID	None
Clear All Touch Regions	134	86	†	None	None
Change Touch Reporting Style	135	87	‡	ReportingType	None
Get Touch Reporting Style	136	88	ˆ	None	Result, ReportingType
Set Dragging Threshold	137	89	‰	Threshold	None
Calibrate Touch Screen	139	8B	‹	None	Result
Load Region File	140	8C	Œ	FileName	Result
Restore Touch Calibration	141	8D	•	None	Result
Set Out of Region Setting	142	8E	Ž	Setting	None
Get Out of Region Setting	143	8F	•	None	Report
Set Region Activate State	146	92	’	RegionID, Enable	Result
Get Region Activate State	147	93	“	RegionID	Enable
Create a Toggle Region	150	96	–	RegionID, X, Y, Width, Height, OffID, OnID	Result
Create a Slider	161	A1	¡	RegionID, X, Y, LT, RB, TrkWidth, TrkHeight, BtnWidth, BtnHeight, TrkID, BtnID, Style	Result
Create a Filled Slider	163	A3	£	RegionID, X, Y, LT, RB, TrkWidth, TrkHeight, BtnWidth, BtnHeight, TrkID, FilID, BtnID, Style	Result
Set Slider Value	166	A6	¦	RegionID, Value	Result
Get Slider Value	167	A7	§	RegionID	Result, Value
Set Toggle State	170	AA	ª	RegionID, State	Result
Get Toggle State	171	AB	«	RegionID	Result, State

Table 48: Output Commands

Name	Dec	Hex	ASCII	Parameters	Response
Set GPO State	73	49	I	Number, Setting	None
Activate Motor	160	A0		Frequency, Duration	None
Set Input Feedback	182	B6	¶	InputOutputType, DownFrequency, UpFrequency	None
Activate Buzzer and Motor	183	B7	·	Frequency, Duration	None
Activate Buzzer	187	BB	»	Frequency, Duration	None
Set Default Buzzer Beep	188	BC	¼	Frequency, Duration	None

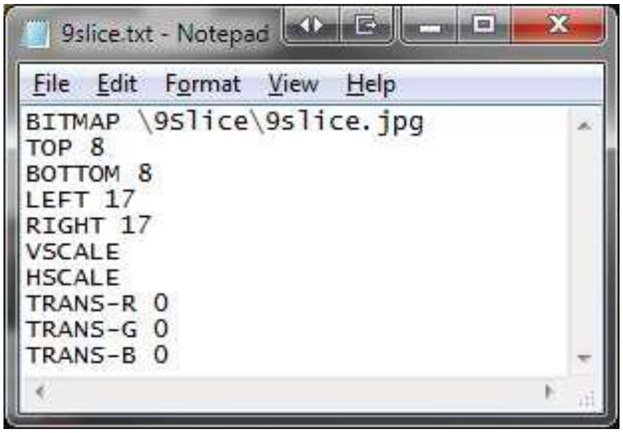
Table 49: Scripts Commands

Name	Dec	Hex	ASCII	Parameters	Response
Run Script File	93	5D	]	FileName	None
Create a Scripted Region	131	83	ƒ	RegionID, X, Y, W, H, UpBitmap, DownBitmap, UpScript, DownScript	None
Create a Scripted Key	138	8A	Š	KeyID, Row, Col, UpScript, DownScript	None
Create a Scripted Toggle Region	162	A2	¢	RegionID, X, Y, Width, Height, OffID, OnID, OffScript, OnScript	Result

3.2 File Examples

9-Slices

The 9-Slice file format is a simple text file that describes how to take a bitmap, and slice it to scale nicely. An example file would be:



```
BITMAP \9slice\9slice.jpg
TOP 8
BOTTOM 8
LEFT 17
RIGHT 17
VSCALE
HSCALE
TRANS-R 0
TRANS-G 0
TRANS-B 0
```

Figure 4: 9-Slice File Example

Each line must start with a keyword, followed by parameters. If a line contains an unrecognized keyword, the line is ignored. The following keywords are defined:

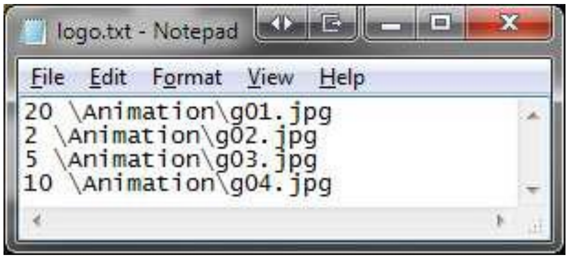
Table 50: 9-slice Keywords

Keyword	Parameters	Description
BITMAP	1	Following the keyword, the bitmap that will be sliced is specified
TOP	1	Specifies how many pixels will be used from the top, for the top slice
BOTTOM	1	Specifies how many pixels will be used from the bottom, for the bottom slice
LEFT	1	Specifies how many pixels will be used from the left, for the left slice
RIGHT	1	Specifies how many pixels will be used from the right, for the right slice
VSCALE	0	If this keyword is present, when the 9-Slice is resized it will stretch the middle left and middle right slices to fill the space required. Without this keyword present, the tile will be repeated from the top down to fill the space.
HSCALE	0	If this keyword is present, when the 9-Slice is resized it will stretch the middle top and middle bottom slices to fill the space required. Without this keyword present, the tile will be repeated from the left to right to fill the space.
TRANS-R	1	The red component of the colour to make transparent in the 9-Slice
TRANS-G	1	The green component of the colour to make transparent in the 9-Slice
TRANS-B	1	The blue component of the colour to make transparent in the 9-Slice

Animations

While the data for animations are stored in the buffer system outlined in the Buffers Section, the actual state of animations are stored in a separate series of animation buffers.

The animation descriptor file is a simple text file, with a series of lines of times to display a frame, and a bitmap to use for that frame. For example:



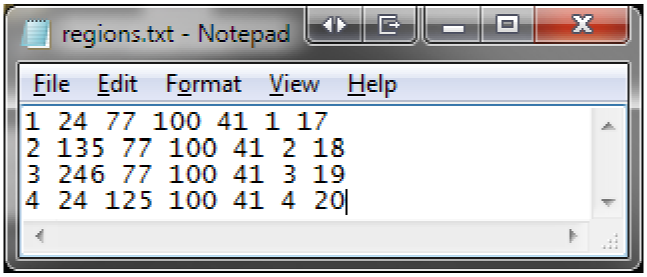
```
20 \Animation\g01.jpg
2 \Animation\g02.jpg
5 \Animation\g03.jpg
10 \Animation\g04.jpg
```

Figure 5: Animation File Example

The above example would define a simple animation with 4 frames. Frame 1 is displayed for 20ms, frame 2 is displayed for 2ms, frame 3 for 5ms, and frame 4 is displayed for 10ms. All file paths must be references with an absolute path from the root.

Region File

Region files can be created using any text editing software. Each line in a region file describes a single touch. There must be no leading blank spaces, only a single space between each field, and no trailing spaces. Bitmap buffers specified must be pre-loaded with desired images. An example of the first row of the calculator demo is shown below.



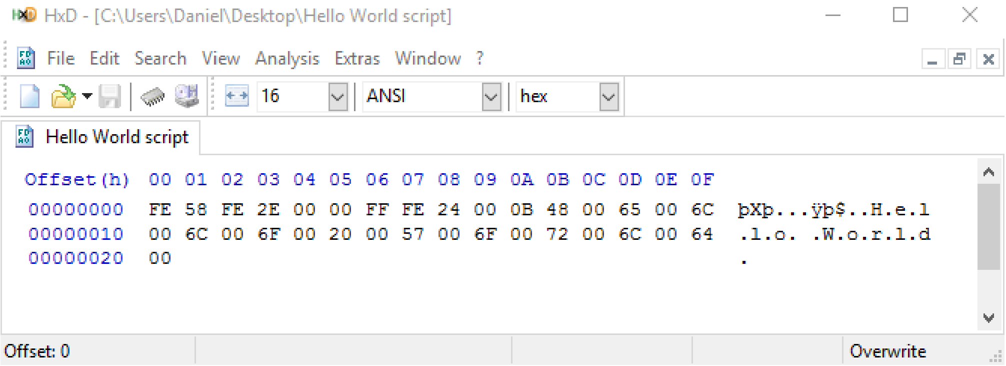
```
1 24 77 100 41 1 17
2 135 77 100 41 2 18
3 246 77 100 41 3 19
4 24 125 100 41 4 20
```

Figure 6: Region File Example

The file above would define four touch regions. The first has an index of 1 is positioned at coordinates (24, 77), a width of 100, and a height of 41. When it is pressed the bitmap in bitmap index 1 will be displayed, and when it is not pressed bitmap 17 will be displayed. Three similar regions follow this one.

Script

Scripts, similar to an AUTOEXEC, can be created by placing the binary stream of values that the module should execute when the script is called. The script below will clear the screen, set font color to blue, and write “Hello World” on the GTT.


HxD - [C:\Users\Daniel\Desktop\Hello World script]
File Edit Search View Analysis Extras Window ?
16
ANSI
hex
Hello World script
Offset: 0
Overwrite

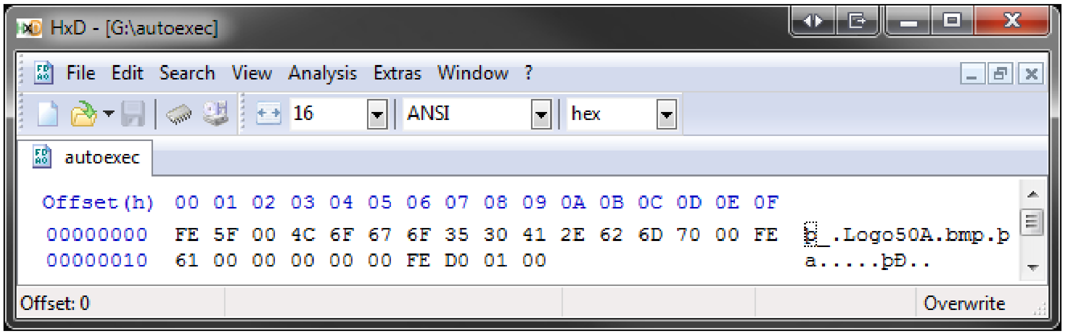
```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  FE 58 FE 2E 00 00 FF FE 24 00 0B 48 00 65 00 6C  þXþ...ÿþ$..H.e.l
00000010  00 6C 00 6F 00 20 00 57 00 6F 00 72 00 6C 00 64  .l.o. .W.o.r.l.d
00000020  00                                               .
```

Please note, if a script executes, and a command is started within the script, however is not completed with the data in the script, the command will wait for data from the serial port to complete the command. After which, the module will return to normal operations.

Autoexec File

In order to create an autoexec file that will run on your GTT, simply place the binary stream of values that the module should execute on startup in the AUTOEXEC. The default autoexec file below, which ships from the factory, loads and displays a start screen before clearing the bitmap buffer.



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  FE 5F 00 4C 6F 67 6F 35 30 41 2E 62 6D 70 00 FE  þ_.Logo50A.bmp.þ
00000010  61 00 00 00 00 00 FE D0 01 00                    a.....þÐ..
```

Figure 7: Autoexec File Example

Please note, if a command is started within the AUTOEXEC, however is not completed with the data in the AUTOEXEC, the command will wait for data from the serial port to complete the command. After which, the module will return to normal operations. The AUTOEXEC file is a special example of the script feature available on the GTT line.

3.3 Memory

Table 51: Valid Memory Card Types

Size	Type
128MB – 2GB	SD
4GB – 32GB	SDHC
64GB – 2TB	SDXC

Table 52: Communication Buffers

Buffer	Size
Data buffer	4kB
FIFO queue Size	16 Byte

Table 53: RAM Allocation

Description	Size
Reserved RAM	~2MB
Buffers	30MB

***Note:** Despite generous buffer sizes, hardware flow control is recommended for all communication.

3.4 Data Types

Common Language Representations

The following table outlines native data types in common programming languages that can be used to represent the data types used in this manual.

Table 54: Data Types with Representations

	ANSI C/C++	C#	Visual Basic
Byte	unsigned char	byte	Byte
Signed Byte	signed char	Sbyte	SByte
Short	unsigned short	ushort	UShort
Signed Short	short	short	Short
Integer	unsigned int	uint	UInteger
Signed Integer	int	int	Integer
String	string	string	String

Table 55: Data Type Descriptions

Byte	Unsigned 8 bit data type that can represent a value from 0 to 255.
Signed Byte	Signed 8 bit data type that can represent a value from -128 to 127.
Short**	Unsigned 16 bit data type can represent values from 0 to 65,536.
Signed Short**	Signed 16 bit data type that can represent values from -32,768 to 32,767.
Integer **	Unsigned 32 bit data type that can represent values from 0 to 4,294,967,295.
Signed Integer**	Signed 32 bit data type that can represent values of -2,147,483,648 to 2,147,483.
String	Strings are a length of bytes terminated by a single null byte. The ASCII character set is used by default, but Unicode or UTF-8 strings may be used where specifically outlined.

****Note:** Transmission of multiple byte values can be set to either big or little endian order.

4 Definitions

9-Slice: Graphic format used to scale bitmaps, usually rectangular, without distorting their geometry. Nine regions define the object center, four corners, and four sides for accurate up or down scaling.

ASCII: American standard code for information interchange used to give standardized numeric codes to alphanumeric characters.

Big Endian: Transmission protocol whereby the most significant byte is transmitted first.

BPS: Bits per second, a measure of transmission speed.

GUI: Graphical user interface.

Hexadecimal: A base 16 number system utilizing symbols 0 through F to represent the values 0-15.

I^2C: Inter-integrated circuit protocol uses clock and data lines to communicate short distances at slow speeds from a master to up to 128 addressable slave devices. A display is a slave device.

Little Endian: Transmission protocol whereby the least significant byte is transmitted first.

LSB: Least significant bit or byte in a transmission, the rightmost when read.

MSB: Most significant bit or byte in a transmission, the leftmost when read.

RS232: Recommended standard 232, a common serial protocol. A low level is -30V, a high is +30V.

RS422: Recommended standard 422, a more robust differential pair serial protocol.

SDA: Serial data line used to transfer data in I^2C protocol. This open drain line should be pulled high through a resistor. Nominal values are between 1K and 10K Ω.

SCL: Serial clock line used to designate data bits in I^2C protocol. This open drain line should be pulled high through a resistor. Nominal values are between 1K and 10K Ω.

TTL: Transistor-transistor logic applied to serial protocol. Low level is 0V while high logic is 5V.

TFT: Thin film transistor, used in reference to a crisp, full-colour LCD technology.

USB: Universal Serial Bus protocol widely used in PCs.

5 Contact

Sales	Support	Design	Online
Phone: 403.229.2737	Phone: 403.229.2737	Phone: 403.229.2737	Purchase: www.matrixorbital.com
Email: sales@matrixorbital.ca	Email: support@matrixorbital.ca	Email: design@matrixorbital.ca	Support: www.matrixorbital.ca