

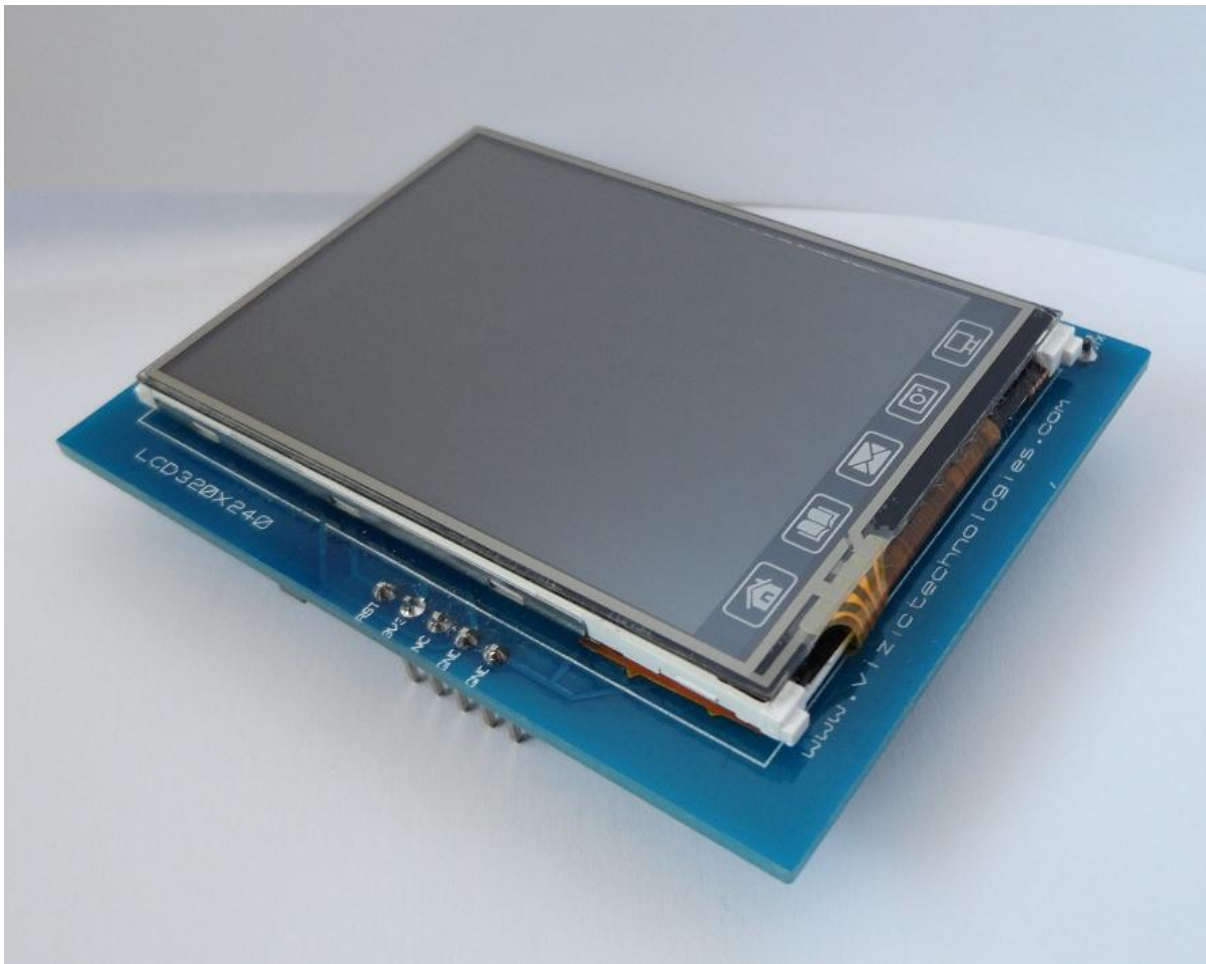


VIZIC
TECHNOLOGIES

SMART GPU
SW VER2

COMMAND SET----Rev 2.0

SMART GPU – Intelligent Embedded Graphics Processor Unit



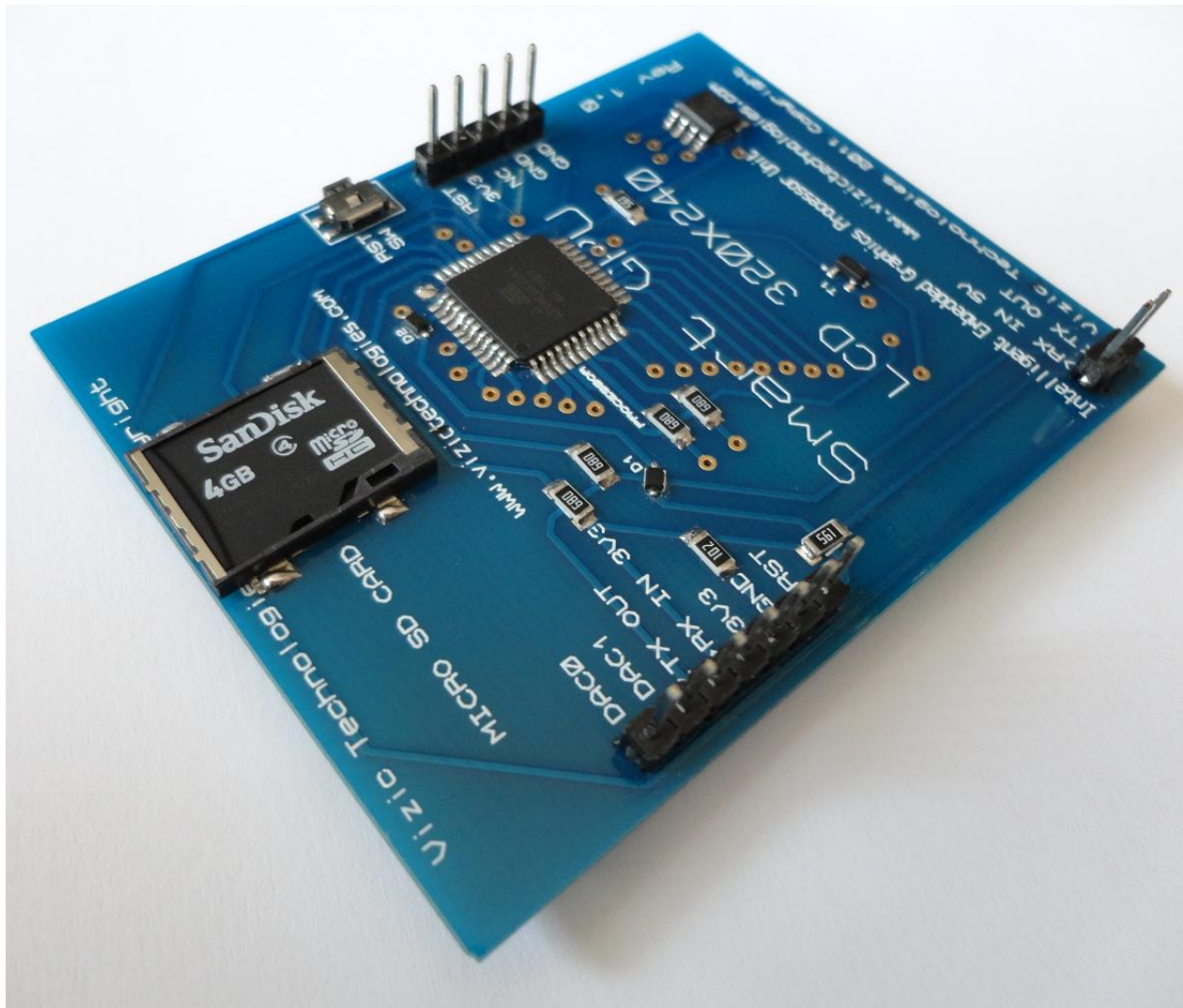


Table of Contents:

Introduction.....	5
Features.....	6
SmartGPU Explained.....	7
1. Host Interface.....	8
1.1 Command Protocol : Flow Control.....	8
1.2 Serial Set-up.....	9
1.3 Power-up and Reset.....	9
1.4 Splash Screen on Power Up.....	10
1.5 Understanding the computer's graphic coordinate system.....	10
2. SMART GPU Command Set software Interface Specification.....	11
2.1 General Commands.....	12
2.1.1 Initialize SMART GPU-55hex.....	13
2.1.2 Set Background Colour - 42hex.....	14
2.1.3 Erase Screen - 45hex.....	15
2.1.4 Display Brightness - 56hex	16
2.1.5 Sleep – 5Ahex.....	17
2.1.6 Display Orientation – 4Fhex.....	18
2.1.7 BaudRate Change – 58hex.....	19
2.2 Graphics Commands.....	20
2.2.1 Put Pixel – 50hex.....	21
2.2.2 Draw Line – 4Chex	22
2.2.3 Draw Rectangle – 52hex.....	23
2.2.4 Draw Circle – 43hex.....	25
2.2.5 Draw Triangle – 54hex.....	27
2.2.6 Draw Image/Icon – 49hex.....	29
2.3 Text Commands.....	31
2.3.1 Set Text Background Colour - 41hex	32
2.3.2 Put Letter – 57hex.....	33
2.3.3 Display String – 53hex.....	35
2.4 Micro SD card Commands.....	37
2.4.1 Image SD – 49hex.....	38
2.4.2 String SD – 53hex.....	41
2.5 FAT Data Management/Data Logger Commands.....	45
2.5.1 Open File – 46hex+4Fhex.....	47
2.5.2 Close File – 46hex+43hex.....	49
2.5.3 Sync File – 46hex+53hex.....	50
2.5.4 Set Pointer – 46hex+50hex.....	51
2.5.5 Read File – 46hex+52hex.....	52
2.5.6 Write File – 46hex+57hex.....	54
2.5.7 List Files – 46hex+48hex.....	55

2.6 Touch Commands.....	56
2.6.1 Get Touch – 47hex	57
2.6.2 Calibrate Touch – 48hex - H ascii (ADVANCED USERS).....	59
2.7 Memory Read and Digital Out Commands.....	60
2.7.1 Memory Read – 4Dhex.....	61
2.7.2 Digital Out Pin– 44hex.....	63
4. Development software tools.....	64
Proprietary Information.....	65
Disclaimer of Warranties & Limitation of Liability.....	65

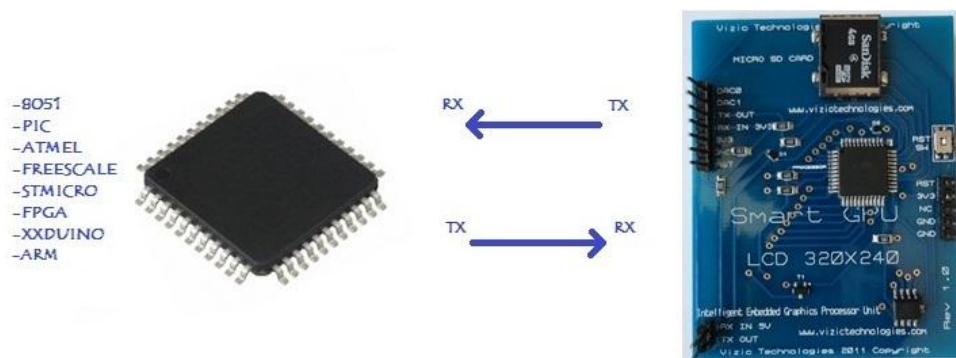
**Commands available only on Smart GPU Software Version 2, all units sold after June 3rd 2012 are software ver 2.*

Introduction:

The SMART GPU-Intelligent Graphics Processor Unit is a powerful easy to use embedded development/professional board with a touch color LCD, and touch controller. It offers a simple yet effective serial interface to any host micro-controller that can communicate via a serial port. All screen related functions are sent using a simple protocol via the serial interface. The SMART GPU allows users to develop their application using their favorite micro-controller or FPGA and software development tools. In short it offers one of the most flexible embedded graphics solutions available.

The SMART GPU processor doesn't need any configuration or programming on itself, it's a slave device that only receives orders, reducing and facilitating dramatically the code size, complexity and processing load on your favorite main processor (8051, PIC, ATMEL, FREESCALE, STMICRO, ARM, CORTEX, any development platform (ARDUINO or similar, FPGA MBED, etc.) or PC(serialport)) of your application.

The next image shows more clearly the roles played by the main processor of your application and the SMARTGPU:



Main Processor:

- main application processing.
- math processing
- I/O processing

VS

SMART GPU Processor:

- Color processing
- Images processing
- SD memory card processing
- Geometry processing
- Text processing
- Touch processing
- Memory management
- And more...

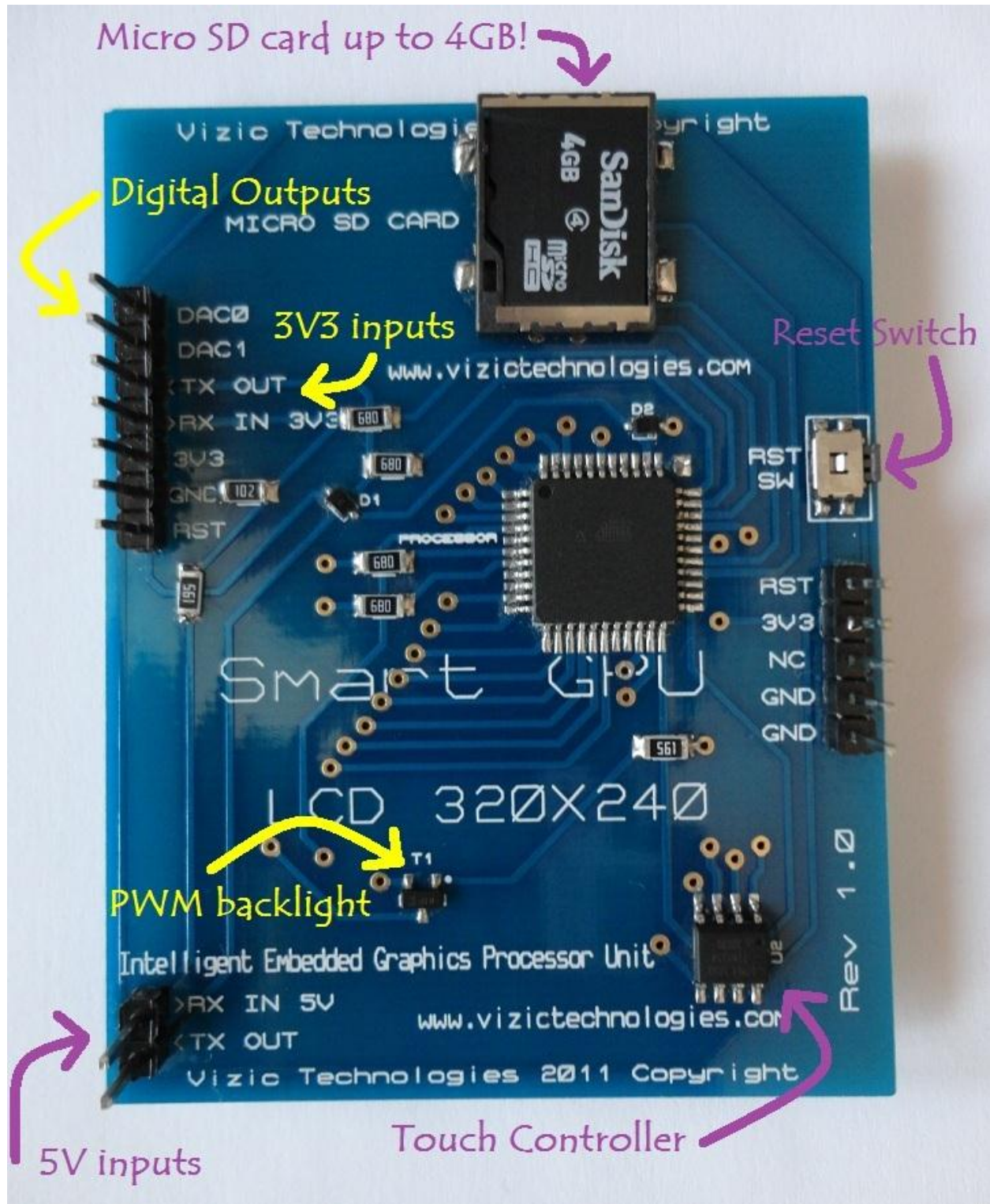
Instead of loading all the Geometry, Images, SD memory access, etc. processing to your main processor, the SMART GPU does all the job and stuff in parallel with your microcontroller or FPGA for you by simple orders or commands.

The main goal of the SMART GPU it's to bring a very easy way to add colour, visual and touch human interfacing to any application or project, without the user having experience in handling LCDs and graphics algorithms. Although it's very easy to use, the SMART GPU it's a low power/very high performance graphics processor, with a microSD card slot supporting up to 32 GB of storage (read/write), and FAT/FAT12/FAT16 or FAT32 universal file System that is compatible with any PC, no special format is needed.

Features:

- 2.4" LCD capable of displaying 262,144 colors.
- Easy 5 pin interface to any host device: **VCC, TX, RX, GND, RESET.**
- On-board uSD/uSDHC memory card adaptor compatible with FAT(windows PC), Support up to **32GB** for storing images and text, **New Data Logger functions read/write (only Soft V2).**
- Integrated Touch screen driver, 10 bit accuracy touch.
- PWM controlled display brightness.
- 5 general purpose Icons on touch
- Sleep mode.
- 2 General purpose Digital Output pins on board
- Baud Rate speed up to 2000000 bps, 8 bits, no parity, 1 stop bit.
- 5V and 3V3 I/O compatible.
- 3V3 power supply.
- External reset switch

SMART GPU EXPLAINED



1.-Host Interface

The SMART GPU is a slave peripheral device and it provides a bidirectional serial interface to a host controller via its UART(Universal Asynchronous Receiver - Transmitter).

Any microcontroller or processor (AVR, PIC, BASICstamp, XXDUINO, 8051, MBED, FPGA, ARM, STmicro, etc) or PC(by serial interface RS232) as host, can communicate to the device over this serial interface from 9600bps up to 2000000bps.

The SMART GPU doesn't need to be configured in any way; it's a plug-and-play device, could be used by students, up to industrial and professional applications, its compatible with any device and existing development board with a UART.

The serial protocol is universal and very easy to implement.

Serial Data Format: 8 Bits, No Parity, 1 Stop Bit.

BaudRate: 9600 bps (default; could be changed).

Serial data is true and not inverted.

1.1 Command Protocol : Flow Control

The SMART GPU Intelligent Graphics Processor Unit is a slave device and all communication and events must be initiated first by the host. Commands consist of a sequence of data bytes beginning with the command/function byte.

When a command is sent from host to the device, this process the command and when the operation is completed, it will always return a response*. The device will send back a single acknowledge byte called the ACK (4Fhex, 'O' ascii), in the case of success, or NAK (46hex, 'F' ascii), in the case of failure or not recognized command.

* Commands having specific responses may send back varying numbers of bytes, depending upon the command and response. It will take the device a certain amount of time to respond, depending on the command type and the operation that has to be performed.

1.2 Serial Set-up

The SMART GPU is configured to be always initialized at a standard **baud rate of 9600 bps**. So the first command that the host sends to the SMART GPU must be at that speed.

Always after any power-up or reset, the SMART GPU must be initialized by sending the uppercase ascii character '**U**' (55hex) at 9600bps. This will initialize all the SMART GPU processor, and when done it will respond with an ACK byte (4Fhex, 'O'ascii).

If the SMART GPU respond with a NAK(46hex, 'F'ascii), Host must try to send the uppercase ascii character '**U**' (55hex) again until a valid ACK is received, meaning this that SMART GPU is ready.

Once the SMART GPU is initialized, user can change the baud rate speed to a total of 8 different speeds up to 2Mbps.

Remember:

The SMART GPU always initializes the micro SD card after a valid 'U' character is received. If a micro SD card is detected the ACK 'O' will be response almost immediately, however if no micro SD card is detected, the ACK 'O' could be delayed while the SMART GPU retries to initialize a micro SD card, however if no micro SD card is detected after several tries, the SMART GPU will send the ACK 'O' and the processor will function normally without the SD card functions.

1.3 Power-up and Reset

When the SMART GPU device comes out of a power up or external reset, a 200ms delay before sending any command must be met, do not attempt to communicate with the module before this period.

If no valid uppercase ascii character '**U**' (55hex) is sent before 6 seconds, the SMART GPU logo will automatically show up, host still can send the uppercase ascii character '**U**' (55hex) to initialize the SMART GPU even if the logo has already appeared.

Remember:

The host transmits the upper case character ('U', 55hex) as the first command so the device to start communication.

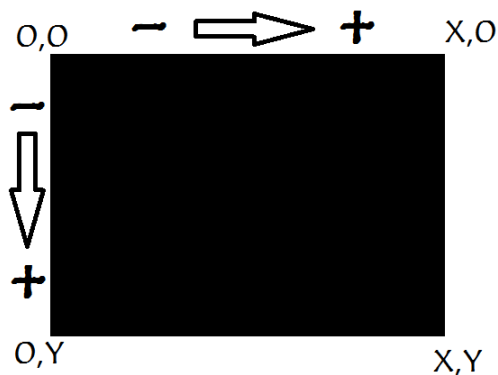
1.4 Splash Screen on Power Up

The SMART GPU will wait up to 6 seconds with its screen in black, for the host to transmit the Initial command ('U', 55hex). If the host has not transmitted this initial command the module will display its splash screen. If the host has transmitted only the initial command and has received a valid ACK, the screen will remain in black. This wait period of the splash screen to appear, is to allow the user initialize the SMART GPU before the welcome screen appears when it is undesired.

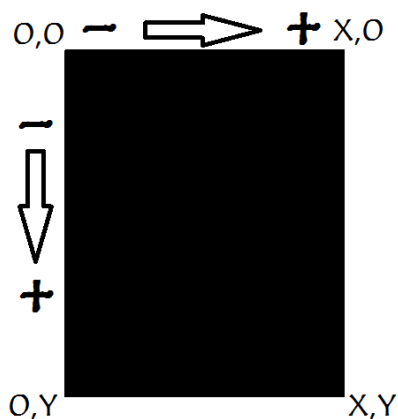
1.5 Understanding the Computer's graphic coordinate system

As well as a computer monitor's coordinate system, the SMART GPU uses the same universal coordinate system, on computer's there's only one positive coordinate quadrant, and there's no negative numbers or points. This quadrant is represented as follows:

The upper left corner is 0,0 if we go right the X values increases, as we go down the Y values increase.



This image shows a **LANDSCAPE** orientation of the screen, the upper left corner is 0,0 (zero,zero). The maximum values of the SMART GPU in LANDSCAPE mode are X:319,Y:239.



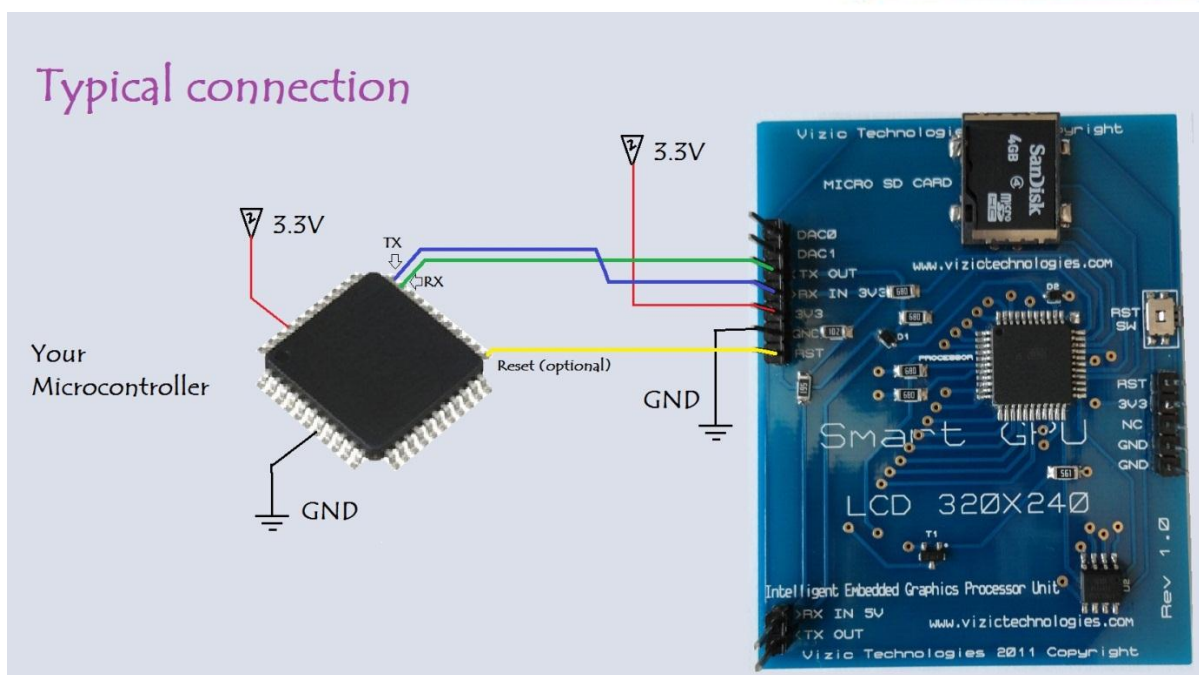
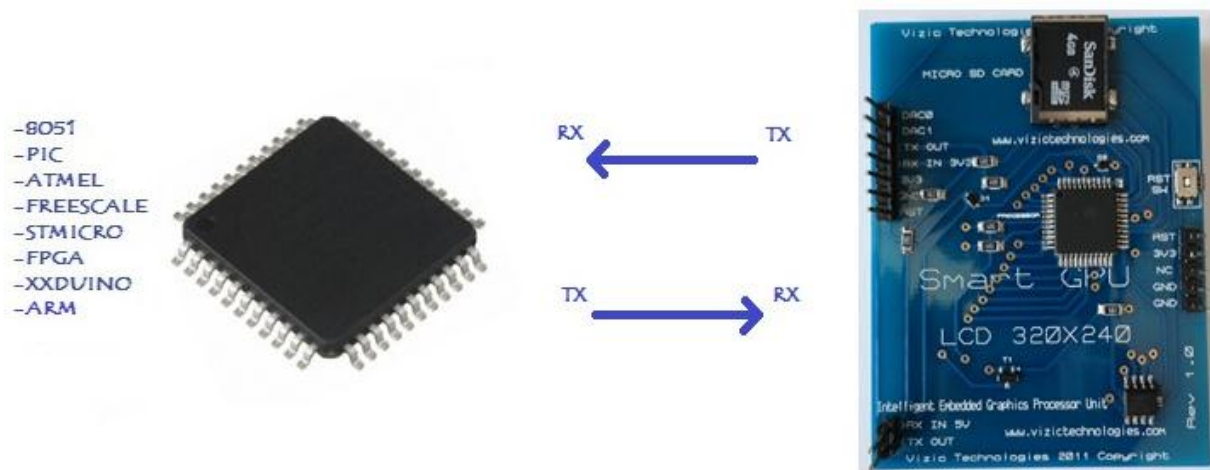
This image shows a **PORTRAIT** orientation of the screen, the upper left corner is 0,0 (zero,zero). The maximum values of the SMART GPU in PORTRAIT mode are X:239,Y:319.

2. SMART GPU Command Set - Software Interface Specification

As mentioned before the command interface between the SMART GPU and the host is via the serial interface UART.

A list of very easy to learn commands provide complete access to all the available functions. Commands and responses can be a single byte or a byte package. All commands always return a response, either a single ACK, or data followed by an ACK.

Remember all commands start with a uppercase letter (ascii).



2.1 General Commands

Briefly Summary of Commands in this section:

- Initialize SMART GPU – 55hex 'U'
- Set Background Colour – 42hex 'B'
- Erase Screen – 45hex 'E'
- Display Brightness – 56hex 'V'
- Sleep – 5Ahex 'Z'
- Display Orientation – 4Fhex 'O'
- BaudRate Change – 58hex 'X'

The colour parameter needed on the Set Background Colour command, consist of 16bits (2 bytes) RGB565:

R4R3R2R1R0G5G4G3 G2G1G0B4B3B2B1B0

That is:

5bits for red, 6 bits for green, 5bits for blue.

High byte colour: R4R3R2R1R0G5G4G3

Low byte colour: G2G1G0B4B3B2B1B0

2.1.1 Initialize SMART GPU - 55hex - U ascii

Commands (host)	1 byte 1.- 0x55 (hex), U (ascii).
Responses (device)	1 byte 1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.
Description	This command is needed only once to initialize communication with the SMART GPU after any power-up or reset, remember to wait at least 200ms after any power up or reset before sending this command.
Example (sent commands)	-55- Initializes SMART GPU. All data is hex.

2.1.2 Set Background Colour - 42hex - B ascii

Commands (host)	3 bytes 1.- 0x42 (hex), B (ascii). 2.- High byte colour. 3.- Low byte colour.
Responses (device)	1 byte 1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.
Description	<p>Command needed to set background, the colour consist of 16bits (2 bytes) RGB565:</p> <p>R4R3R2R1R0G5G4G3G2G1G0B4B3B2B1B0</p> <p>That is: 5bits for red, 6 bits for green, 5bits for blue.</p> <p>High byte : R4R3R2R1R0G5G4G3 Low byte : G2G1G0B4B3B2B1B0</p> <p>Once this command is sent, each time that the device receives the Erase Command, the screen will be draw with this background colour. Default background colour on reset or power on is black.</p>
Example (sent commands)	<p>-42,FF,FF- Sets background to white (FFFF). -42,F8,00- Sets background to red (F800). -42,00,1F- Sets background to blue (001F).</p> <p>All data is in hex.</p>

2.1.3 Erase Screen - 45hex - E ascii

Commands (host)	1 bytes
	1.- 0x45 (hex), E (ascii).
Responses (device)	1 byte
	1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.
Description	Command needed to erase the entire screen, the display will be draw with the background colour if it is set before, if not, default background colour on reset or power on is black.
Example (sent commands)	-45- Erase screen with set background colour. All data is in hex.

2.1.4 Display Brightness - 56hex - V ascii

Commands (host)	2 bytes
	1.- 0x56 (hex), V (ascii). 2.- Brightness value (0-128) (0 hex-80 hex).
Responses (device)	1 byte
	1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.
Description	Command needed to adjust the display brightness, 0(0hex) stands for none, 128(80hex) stands for maximum brightness. Default brightness on reset or power on is 128(80hex).
Example (sent commands)	-56,00- Set minimum brightness (led off). -56,80- Set maximum brightness. All data is in hex.

2.1.5 Sleep – 5Ahex - Z ascii

Commands (host)	2 bytes
	1.- 0x5A (hex), Z (ascii). 2.- Sleep On: 01(hex). Sleep Off: 00(hex).
Responses (device)	1 byte
	1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.
Description	Command needed to set sleep mode. It takes 150ms to get in/out sleep mode after command is accepted. During sleep mode the screen turns completely white, the LCD oscillator and voltage generator turns off, but the memory is conserved. To save more power combine this command with Display Brightness set to zero.
Example (sent commands)	-5A,01- Set sleep mode On. -5A,00- Set sleep mode Off. All data is in hex.

2.1.6 Display Orientation – 4Fhex - O ascii

Commands (host)	2 bytes
	1.- 0x4F (hex), O (ascii). 2.- HorizontalL(landscape): 00(hex) or VerticalL(portrait): 01(hex) or HorizontalR(landscape): 02(hex) or VerticalT(portrait): 03(hex).
Responses (device)	1 byte
	1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.
Description	Command needed to set display orientation mode, landscape or portrait. Default display orientation mode on reset or power on is HorizontalL 00(hex).
Example (sent commands)	-4F,01- Set VerticalL mode. -4F,02- Set HorizontalR mode. All data is in hex.

2.1.7 BaudRate Change – 58hex - X ascii

Commands (host)	2 bytes 1.- 0x58 (hex), X (ascii). 2.- 9600 bps : 00(hex) or 19200 bps : 01(hex) or 57600 bps : 02(hex) or 115200 bps : 03(hex) or 256000 bps : 04(hex) or 500000 bps : 05(hex) or 1000000 bps : 06(hex) or 2000000 bps : 07(hex).
Responses (device)	1 byte fail, 2 bytes success 1.- 0x4F (hex), O (ascii) – success ACK at actual baud rate. Delay 500ms. 2.- 0x4F (hex), O (ascii) – success ACK at new baudrate. or 1.- 0x46 (hex), F (ascii) – fail NAK at actual baud rate.
Description	<p>Command needed to set a different baud rate, the command sent by the host must be at the actual baud rate that is being used; if the command is invalid a NAK will be responded by the SMART GPU and the baud rate will not be modified.</p> <p>If the command is accepted an ACK will be received at the actual baud rate then SMART GPU will change to the new baud rate during 500ms and then it will send another ACK at the new baud rate selected.</p> <p>Only when an ACK has been received by the host, the next commands must be sent at the new baud rate defined.</p> <p>Default reset - power on baud rate is 9600.</p>
Example (sent commands)	-58,01- Set 19200 bps baud. -58,03- Set 115200 bps baud. -58,02- Set 57600 bps baud. -58,07- Set 2000000 bps baud. All data is in hex.

2.2 Graphics Commands

Briefly Summary of Commands in this section:

- Put Pixel – 50hex 'P'
- Draw Line – 4Chex 'L'
- Draw Rectangle – 52hex 'R'
- Draw Circle – 43hex 'C'
- Draw Triangle – 54hex 'T'
- Draw Image/Icon – 49hex 'I'

The colour parameter needed on all of those commands, consist of 16bits (2 bytes) RGB565:

R4R3R2R1R0G5G4G3 G2G1G0B4B3B2B1B0

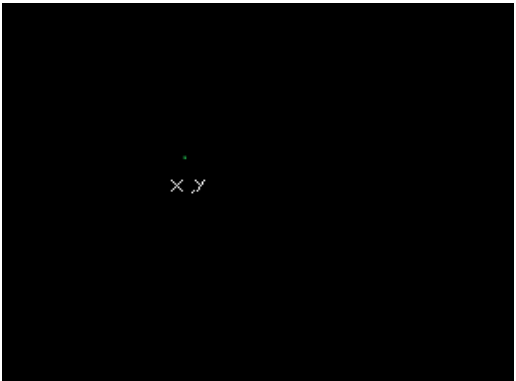
That is:

5bits for red, 6 bits for green, 5bits for blue.

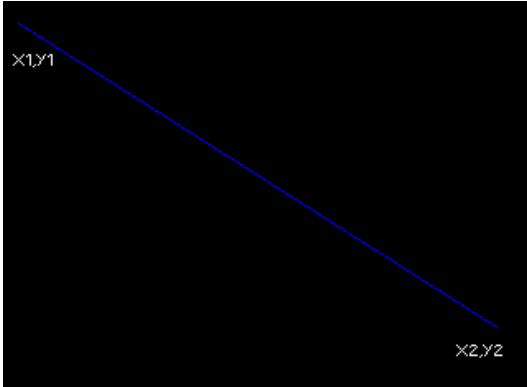
High byte colour: R4R3R2R1R0G5G4G3

Low byte colour: G2G1G0B4B3B2B1B0

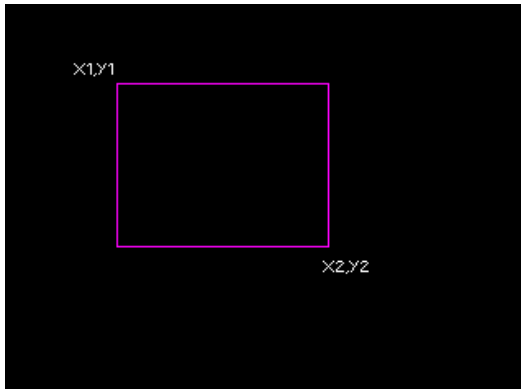
2.2.1 Put Pixel – 50hex - P ascii

Commands (host)	7 bytes
	1.- 0x50 (hex), P (ascii). 2.- X high byte. 3.- X low byte. 4.- Y high byte. 5.- Y low byte. 6.- High byte colour. 7.- Low byte colour.
Responses (device)	1 byte
	1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.
Description	This command draws a simple dot on the screen at the given X(16bit) and Y(16bit) coordinates. Colour format is the same RGB565.
Example (sent commands)	<p>-50,00,32,00,3C,07,E0- Put a Green Pixel at X:50(dec), Y:60(dec).</p>  <p>All data is in hex. Note: Maximum X or Y acceptable size value depend on display orientation.</p>

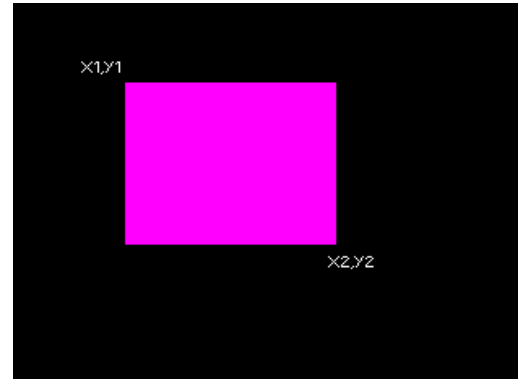
2.2.2 Draw Line – 4Chex - L ascii

Commands (host)	11 bytes
	1.- 0x4C (hex), L (ascii). 2.- X1 high byte. 3.- X1 low byte. 4.- Y1 high byte. 5.- Y1 low byte. 6.- X2 high byte. 7.- X2 low byte. 8.- Y2 high byte. 9.- Y2 low byte. 10.- High byte colour. 11.- Low byte colour.
Responses (device)	1 byte
	1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.
Description	This command draws a simple line on the screen with the two given points: X1(16bit), Y1(16bit) and X2(16bit),Y2(16bit). Colour format is the same RGB565.
Example (sent commands)	<p>-4C,00,0A,00,0F,01,2C,00,C8,00,1F- Draws a Blue line from X1:10(dec),Y1:15(dec) to X2:300(dec),Y2:200(dec).</p>  <p>All data is in hex. Note: Maximum Xs or Ys acceptable size values depend on display orientation.</p>

2.2.3 Draw Rectangle – 52hex - R ascii

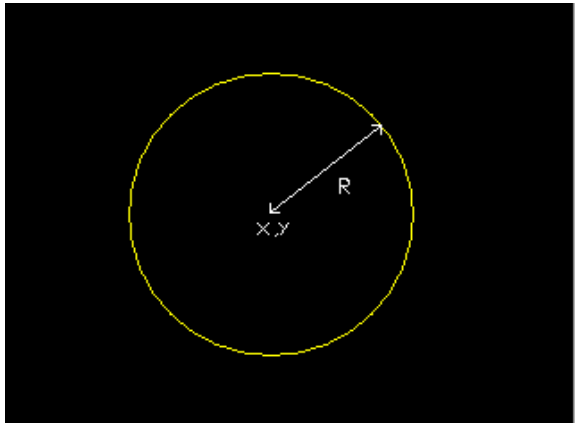
Commands (host)	12 bytes
	1.- 0x52 (hex), R (ascii). 2.- X1 high byte. 3.- X1 low byte. 4.- Y1 high byte. 5.- Y1 low byte. 6.- X2 high byte. 7.- X2 low byte. 8.- Y2 high byte. 9.- Y2 low byte. 10.- High byte colour. 11.- Low byte colour. 12.- Fill: 00(hex) No Fill Geometry or 01(hex) Fill Geometry.
Responses (device)	1 byte
	1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.
Description	This command draws a simple rectangle on the screen with the two given points: X1(16bit), Y1(16bit) and X2(16bit), Y2(16bit). Colour format is the same RGB565.
Example (sent commands)	<p>-52,00,46,00,32,00,C8,00,96,F8,1F,00- Draws a no filled purple rectangle from X1:70(dec),Y1:50(dec) to X2:200(dec), Y2:150(dec).</p> 

-52,00,46,00,32,00,C8,00,96,F8,1F,01- Draws a filled purple rectangle from X1:70(dec),Y1:50(dec) to X2:200(dec),Y2:150(dec).

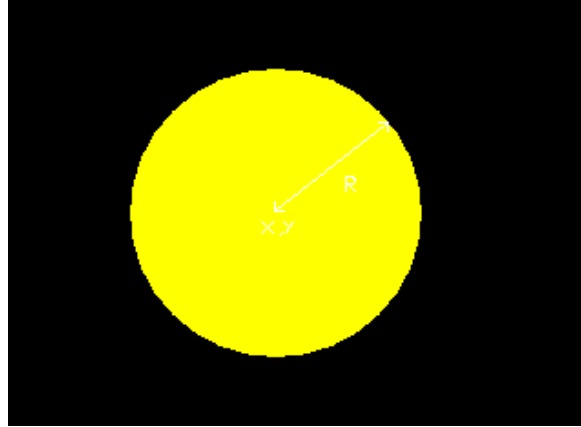


All data is in hex. **Note:** Maximum Xs or Ys acceptable size values depend on display orientation.

2.2.4 Draw Circle – 43hex - C ascii

Commands (host)	10 bytes
	1.- 0x43 (hex), C (ascii). 2.- X high byte. 3.- X low byte. 4.- Y high byte. 5.- Y low byte. 6.- Radius high byte. 7.- Radius low byte. 8.- High byte colour. 9.- Low byte colour. 10.- Fill: 00(hex) No Fill Geometry or 01(hex) Fill Geometry.
Responses (device)	1 byte
	1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.
Description	<p>This command draws a simple circle on the screen with center on the given point: X(16bit),Y(16bit) and RADIUS(16bit) value. Colour format is the same RGB565.</p> <p><i>Radius value must be always different than zero.</i></p>
Example (sent commands)	<p>-43,00,96,00,78,00,50,FF,E0,00- Draws a no filled yellow circle with center X:150(dec),Y:120(dec) and RADIUS:80(dec).</p> 

-43,00,96,00,78,00,50,FF,E0,01- Draws a filled yellow circle with center X:150(dec),Y:120(dec) and RADIUS:80(dec).

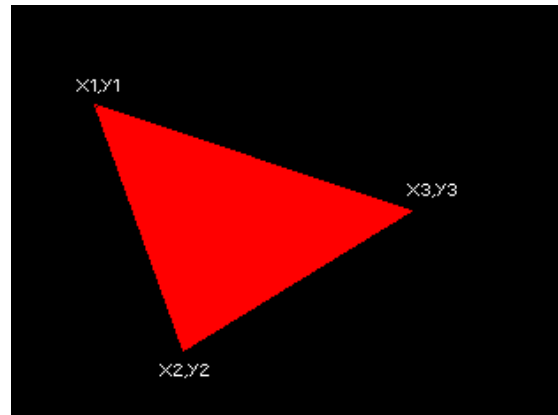


All data is in hex. **Note:** Maximum Xs or Ys acceptable size values depend on display orientation.

2.2.5 Draw Triangle – 54hex - T ascii

Commands (host)	16 bytes
	1.- 0x54 (hex), T (ascii). 2.- X1 high byte. 3.- X1 low byte. 4.- Y1 high byte. 5.- Y1 low byte. 6.- X2 high byte. 7.- X2 low byte. 8.- Y2 high byte. 9.- Y2 low byte. 10.- X3 high byte. 11.- X3 low byte. 12.- Y3 high byte. 13.- Y3 low byte. 14.- High byte colour. 15.- Low byte colour. 16.- Fill: 00(hex) No Fill Geometry or 01(hex) Fill Geometry.
Responses (device)	1 byte
	1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.
Description	This command draws a simple triangle on the screen with the given points: X1(16bit),Y1(16bit), X2(16bit),Y2(16bit) and X3(16bit),Y3(16bit). Colour format is the same RGB565.
Example (sent commands)	-54,00,32,00,3C,00,64,00,C8,00,E6,00,78,F8,00,00- Draws a no filled red triangle with given points: X1:50(dec),Y1:60(dec), X2:100(dec),Y2:200(dec) and X3:230(dec),Y3:120(dec). <div data-bbox="784 1430 1338 1837"> </div>

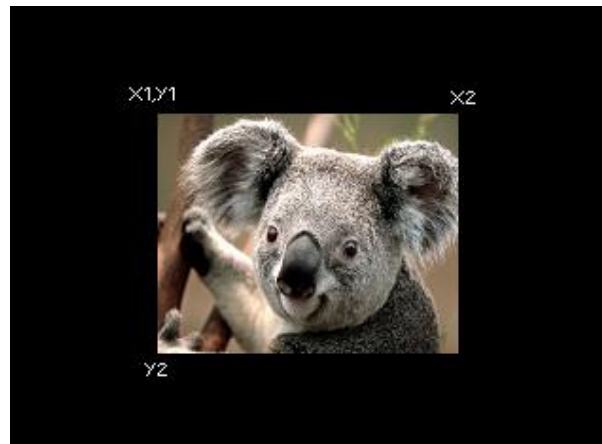
-54,00,32,00,3C,00,64,00,C8,00,E6,00,78,F8,00,01-
Draws a filled red triangle with given points:
X1:50(dec),Y1:60(dec), X2:100(dec),Y2:200(dec) and
X3:230(dec),Y3:120(dec).



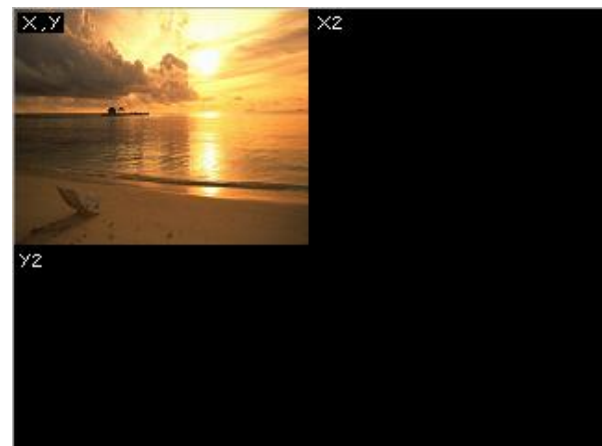
All data is in hex. **Note:** Maximum Xs or Ys acceptable size values depend on display orientation.

2.2.6 Draw Image/Icon – 49hex - I ascii

Commands (host)	10 bytes + pixel number x2
	<p>1.- 0x49 (hex), I (ascii). 2.- 0x00(hex). 3.- X1 high byte. 4.- X1 low byte. 5.- Y1 high byte. 6.- Y1 low byte. 7.- X2 high byte. 8.- X2 low byte. 9.- Y2 high byte. 10.-Y2 low byte. Send now pixel by pixel: 11.- High byte colour pixel 1. 12.- Low byte colour pixel 1. 13.- High byte colour pixel 2. 14.- Low byte colour pixel 2. Last- High byte colour pixel $n((X2-X1)*(Y2-Y1))$. Last- Low byte colour pixel $n((X2-X1)*(Y2-Y1))$.</p> <p><i>Remember: each pixel is composed by two bytes following the same RGB565 convention.</i></p>
Responses (device)	1 byte
	<p>1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.</p> <p><i>Any of those two commands will not be sent until the host finish to send all the width$(X2-X1)*height(Y2-Y1)$ pixels of the image or icon.</i></p>
Description	<p>This command draws an icon/ image on the screen starting at the given points(top left corner): X1(16bit),Y1(16bit) and (bottom right corner)X2(16bit),Y2(16bit).Colour format for each pixel is the same RGB565.</p>
Example (sent commands)	<p>-49,00,00,50,00,3C,00,EF,00,B3, Pix1H,Pix1L,Pix2H,Pix2L,...,PixNH,PixNL- Draws an image of 160x120 pixels with top left corner: X1:80(dec),Y1:60(dec) and bottom right corner: X2:239(dec), Y2:179(dec).</p>



-49,00,00,00,00,00,00,9F,00,77,
Pix1H,Pix1L,Pix2H,Pix2L,...,PixNH,PixNL-
Draws an image of 160x120 pixels with top left
corner: X1:0(dec),Y1:0(dec) and bottom right
corner: X2:159(dec), Y2:119(dec).



All data is in hex. **Note: See that X2 is the sum of X1+(width in pixels - 1), the -1 in the pixels is because pixel number 0 also count. Also Y2 is the sum of Y2+(height in pixels - 1) because the same reason as X2.**

User must be careful not to exceed the 320x240pixels(depending orientation) of the screen in order to avoid unwanted distorted images.

Maximum X1, Y1, X2, Y2 acceptable size values depend on display orientation.

2.3 Text Commands

The SMART GPU is capable of generate 8 different sizes of fonts, with transparent or colour background. Also those 8 fonts are a little different from each other.

The command Display String requires two points that forms an imaginary text box that helps delimit the text writing to certain area, it's helpful when the user only wants to display text only on some area of the screen.

Briefly Summary of Commands in this section:

- Set Text Background – 41hex 'A'
- Put Letter – 57hex 'W'
- Display String – 53hex 'S'

The colour parameter needed on all those commands, consist of 16bits (2 bytes) RGB565:

R4R3R2R1R0G5G4G3 G2G1G0B4B3B2B1B0

That is:

5bits for red, 6 bits for green, 5bits for blue.

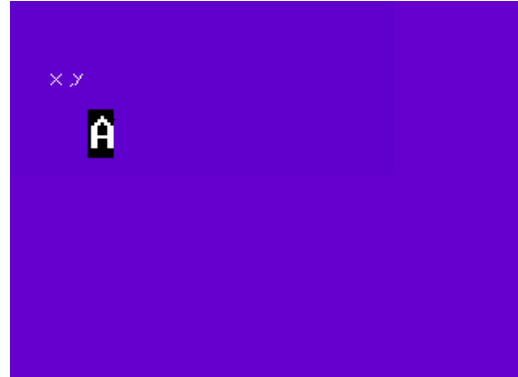
High byte colour: R4R3R2R1R0G5G4G3

Low byte colour: G2G1G0B4B3B2B1B0

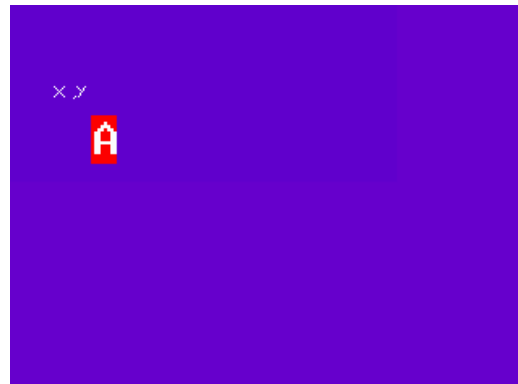
2.3.1 Set Text Background Colour - 41hex - A ascii

Commands (host)	3 bytes 1.- 0x41 (hex), A (ascii). 2.- High byte colour. 3.- Low byte colour.
Responses (device)	1 byte 1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.
Description	<p>Command needed to set text background, the colour consist of 16bits (2 bytes) RGB565:</p> <p>R4R3R2R1R0G5G4G3G2G1G0B4B3B2B1B0 That is: 5bits for red, 6 bits for green, 5bits for blue. High byte : R4R3R2R1R0G5G4G3 Low byte : G2G1G0B4B3B2B1B0</p> <p>Once this command is sent, if the user sends a letter or string and select colour background byte, this new colour will be at the back of text.</p> <p>Default text background colour on reset or power on is black.</p>
Example (sent commands)	-41,FF,FF- Text background to white (FFFF). -41,F8,00- Text background to red (F800). -41,00,1F- Text background to blue (001F). All data is in hex.

-57,00,32,00,46,FF,FF,07,01,41 - Draws a white(0xFFFF), 'A' letter font 07, with top left corner at X:50(dec),Y:70(dec), with colour background (default background colour).



-57,00,32,00,46,FF,FF,07,01,41 - Draws a white(0xFFFF), 'A' letter font 07, with top left corner at X:50(dec),Y:70(dec), with colour background (previously set as red).



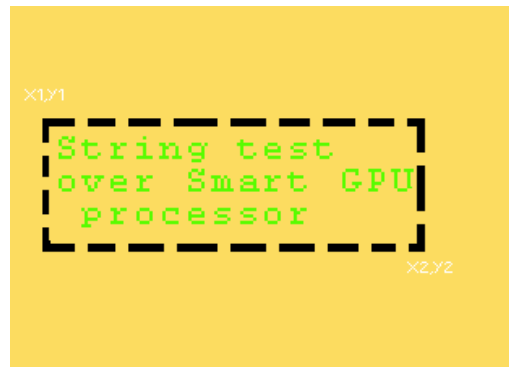
All data is in hex. **Note:** Maximum X or Y acceptable size values depend on display orientation.

Example (sent commands)

-53,00,00,1E,00,50,01,04,00,C8,07,E0,05,00,text,00-
 Draws a green(0x07E0) string, font 05, with top left corner at X1:30(dec), Y1:80(dec), and bottom right corner X2:260(dec), Y2:200(dec), with transparent background.

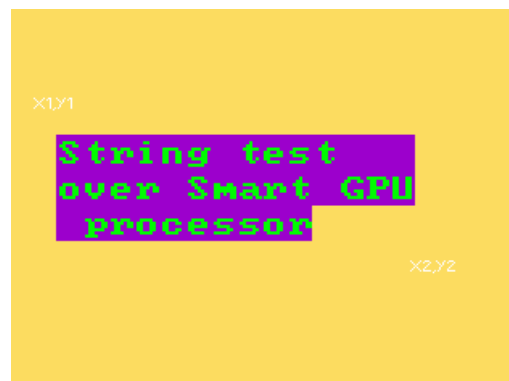
text= String over Smart GPU processor

Note: the black dotted box is just to exemplify the imaginary defined text box, it is not actually drawn.



-53,00,00,1E,00,50,01,04,00,C8,07,E0,06,01,text,00-
 Draws a green(0x07E0) string, font 06, with top left corner at X1:30(dec), Y1:80(dec), and bottom right corner X2:260(dec), Y2:200(dec), with colour background(previously set).

text= String over Smart GPU processor



All data is in hex. **Note:** Maximum X or Y acceptable size values depend on display orientation.

2.4 micro SD card Commands

The SMART GPU, unlike other graphic development tools on the market, is the only embedded graphic processor capable of managing files directly in FAT/FAT12/FAT16 or FAT32 file systems without any special program/interface or micro SD rare formats. Fully compatible with any PC.

Smart GPU can manage images and text files, so any file with .bmp or .txt extension will be easily opened as text(.txt) or image(.bmp) respectively.

Smart GPU Soft Ver2 can create-write-read any type of files.

A maximum of 32GBs micro SD memory card is supported, allowing to store thousands of full screen images or thousands of text files.

Briefly Summary of Commands in this section:

- Image SD – 49hex 'I'
- String SD – 53hex 'S'

A complete tutorial on how to load images and text to the SD card is explained on this PDF file at the SD card file management section.

The colour parameter needed on all those commands, consist of 16bits (2 bytes) RGB565:

R4R3R2R1R0G5G4G3 G2G1G0B4B3B2B1B0

That is:

5bits for red, 6 bits for green, 5bits for blue.

High byte colour: R4R3R2R1R0G5G4G3

Low byte colour: G2G1G0B4B3B2B1B0

2.4.1 Image SD – 49hex - I ascii

Commands (host)	6 bytes + image name + 1byte(NULL)
	1.- 0x49 (hex), I (ascii). 2.- 0x53 (hex), S (ascii). Means Image from SD. 3.- X high byte.(left corner) 4.- X low byte. (left corner) 5.- Y high byte. (top corner). 6.- Y low byte. (top corner). 7.- up to N (file name). N+1.- 0x00 (hex) NULL ascii.
Responses (device)	1 byte
	1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.
Description	<p>This command calls an image stored on the micro SD card and displays it with the give point: X(16bit), Y(16bit) as top left corner.</p> <p>If the image is 320x240 pixels this point must be 0,0 if not the image won't fit on the screen.</p> <p>Any size of image could be called, however user is responsively that the image fits on the screen with the X,Y top left corner adjustment.</p> <p>The file name must be 8 characters or less, not including the .bmp extension. Only numbers and letters are recommended to name the file, some special characters are not allowed.</p> <p>Always a NULL character (0x00)hex must follow the last character of the file name, in order to indicate to SMART GPU the end of this file name.</p>
Example (sent commands)	-49,53,00,00,00,00,4B,4F,41,4C,41,00- Opens the 320x240pixels “KOALA.bmp” file with top left corner at X:00(dec), Y:00(dec) and displays it on the screen.



-49,53,00,00,00,00,4A,65,6C,6C,79,33,32,30,00-
Opens the 160x120 pixels “Jelly320.bmp” file with
top left corner at X:00(dec), Y:00(dec) and displays
it on the screen.



-49,53,00,50,00,3C,4A,65,6C,6C,79,33,32,30,00-
Opens the 160x120 pixels “Jelly320.bmp” file with
top left corner at X:80(dec), Y:60(dec) and displays
it on the screen.



-49,53,00,50,00,3C,66,72,61,63,74,61,6C,00-
Opens the 160x120 pixels “fractal.bmp” file with
top left corner at X:80(dec), Y:60(dec) and displays
it on the screen.



An Image of size 160x120 for example, could not be called to be displayed on X>160(dec) and Y>120(dec), because it won't fit on the screen, distorted image will appear. As mentioned before, user is responsible of calling images with top left corners that warrantee that the image will fit on the display.

All data is in hex. **Note:** Maximum X or Y acceptable size values depend on display orientation and image file size on pixels.

2.4.2 String SD – 53hex - S ascii

Commands (host)	18 bytes + text file name + 1byte(NULL)
	1.- 0x53 (hex), S (ascii). 2.- 0x53 (hex), S (ascii). Means text from SD. 3.- X1 high byte. 4.- X1 low byte. 5.- Y1 high byte. 6.- Y1 low byte. 7.- X2 high byte. 8.- X2 low byte. 9.- Y2 high byte. 10.- Y2 low byte. 11.- High byte string colour. 12.- Low byte string colour. 13.- Font Size: 00(hex)-0x07(hex). 14.- Letter Back: 00(hex) transparent. 01(hex) background colour. 15.- Start Character (position) high byte. 16.- Start Character (position) low byte. 17.- Characters to Read (from position) high byte. 18.- Characters to Read (from position) low byte. 19.- up to N (file name). N+1.- 0x00 (hex) NULL ascii.
Responses (device)	1 byte
	1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.
Description	<p>This command calls a text file stored on the micro SD card and displays the contained text on an imaginary text box created with the given points: X1(16bit), Y1(16bit) as top left corner and X2(16bit), Y2(16bit) as bottom right corner.</p> <p>The text or string is automatically adjusted to fit in this defined text box, if the text is longer to fit in the text box, the string is rewritten to the top left corner.</p> <p>The Start Character parameter means, the number of bytes that will be jumped from the start of file, including spaces. If zero (0x0000)hex, the file will be read from the beginning.</p> <p>The Characters to Read parameter means, the number of bytes that will be read from the Start character position. When this Characters to read parameter is set to zero (0x0000)hex, all the file contents will be read from the given Start Character position parameter.</p>

The maximum number of characters to read is 1900 bytes (characters) in one call.

Colour format is the same RGB565.

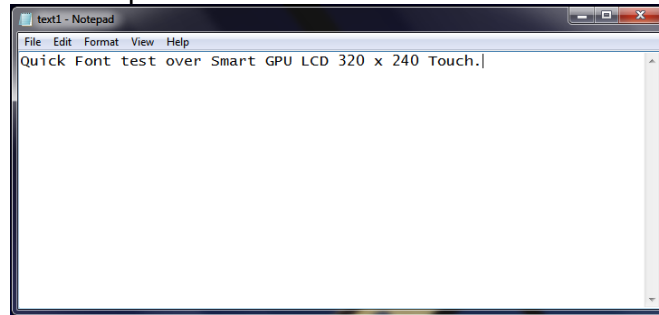
The file name must be 8 characters or less, not including the .txt extension. Only numbers and letters are recommended to name the file, some special characters are not allowed.

Always a NULL character (0x00)hex must follow the last character of the file name, in order to indicate to SMART GPU the end of this file name.

Example (sent commands)

The next examples considered a .txt file previously stored on the micro SD memory card, named “text1.txt”.

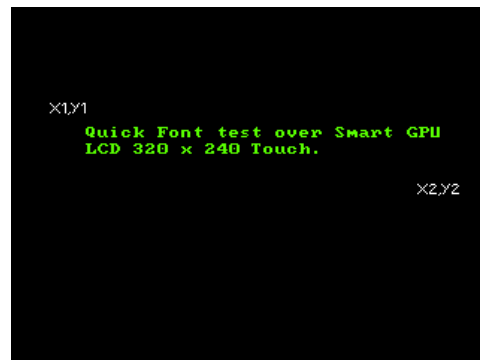
Here's a picture of the contents of the .txt file:



Example 1:

-53,53,00,32,00,50,01,2C,00,C8,07,E0,02,00,00,00,00,00,74,65,78,74,31,00-

Opens the “text1.txt” file of the SDcard with top left corner at X1:50(dec), Y1:80(dec) and bottom right corner at X2:300(dec), Y2:200(dec), with green colour, font #2, no background colour, Start Character 0x0000, Characters to Read 0x0000(means all the file), and displays it on the screen.



Example 2:

-53,53,00,32,00,50,01,2C,00,C8,FF,FF,05,00,00,00,00,1E
,74,65,78,74,31,00-

Opens the same “text1.txt” file of the SDcard with top left corner at X1:50(dec), Y1:80(dec) and bottom right corner at X2:300(dec), Y2:200(dec), with white colour, font #5, no background colour, Start Character 0x0000, Characters to Read 30(dec) 0x001E(hex), and displays it on the screen.



Example 3:

-53,53,00,32,00,50,01,2C,00,C8,FF,FF,05,00,00,0B,00,0E
74,65,78,74,31,00-

Opens the same “text1.txt” file of the SDcard with top left corner at X1:50(dec), Y1:80(dec) and bottom right corner at X2:300(dec), Y2:200(dec), with white colour, font #5, no background colour, Start Character 11(dec) 0x000B(hex), Characters to Read 14(dec) 0x000E(hex), and displays it on the screen.



Just Remember, the **Start Character** is like the pointer of the bytes of the .txt file, the **Characters to Read** is the number of positions that will be read from the **Start Character** pointer position of the .txt file.

All data is in hex. **Note:** Maximum X or Y acceptable size values depend on display orientation.

2.5 FAT Data Management/Data Logger Commands

The new SMART GPU Software version 2 boards include new Data management functions, create files, read files, write files, etc. All units sold after 10 June 2012, are Software Ver 2.

Those new Data Logger functions enable full possibilities with the Smart GPU, as it is now more complete than ever. User can create easy and advanced Data Logger + graphic applications.

Note that commands on all of this section always respond 2 types of ACKs, the only command that respond a single ACK as almost all the Smart GPU graphic commands is "LIST FILES".

Never remove micro SD card during "write" operations on micro SD card, data could be corrupted and damaged.

Briefly Summary of Commands in this section (commands only available on Soft Ver 2):

- Open File –46hex 'F' + 4Fhex 'O'
- Close File –46hex 'F' + 43hex 'C'
- Sync File –46hex 'F' + 53hex 'S'
- Set Pointer –46hex 'F' + 50hex 'P'
- Read File –46hex 'F' + 52hex 'R'
- Write File –46hex 'F' + 57hex 'W'
- List Files –48hex 'H'

File ACK List:

As Mentioned before, those FAT Data Management commands always respond 2 ACKs, first ACK is for File Operation, and second ACK is for command success.

The next list of bytes, are the possible ACKs that could be obtained from File Operation:

Byte received	Meaning	Description
0x00 (hex)	OK	Success.
0x01 (hex)	DISK ERROR	Hard error in low level disk I/O layer.
0x02 (hex)	INTERNAL ERROR	Assertion failed.
0x03 (hex)	NOT READY	Physical drive cannot work.
0x04 (hex)	NO FILE	Could not find the file.
0x05 (hex)	NO PATH	Could not find the path.
0x06 (hex)	INVALID NAME	Path name invalid format.
0x07 (hex)	DENIED	Access denied or full directory.
0x08 (hex)	EXIST	Access denied to prohibited access.
0x09 (hex)	INVALID OBJECT	File object is invalid.
0x0A (hex)	WRITE PROTECTED	Physical drive is write protected.
0x0C (hex)	NOT ENABLED	The volume has no work area.
0x0D (hex)	NO FILESYSTEM	There's no valid FAT volume.
0x13 (hex)	INVALID PARAMETER	Given parameters are invalid.

2.5.1 Open File – 46hex - ‘F’ ascii + 4Fhex – ‘O’ ascii

Commands (host)	3 bytes + file name + “.ext” + 1byte(NULL) 1.- 0x46 (hex), F (ascii). 2.- 0x4F (hex), O (ascii). 3.- Open Mode: 0x01 (hex)- Read Only 0x02 (hex)- Write Only 0x03 (hex)- Read +Write 0x04((hex)- Create New 0x10 (hex)- Open Always 0x08((hex)- Create Always 4.- up to N (file name). N+1.- ‘.’ + extension- example “.txt” N+5.- 0x00 (hex) NULL ascii.
Responses (device)	1 byte File ACK + 1 byte Command ACK 1.- Refer to “File ACK List”. 2.- 0x4F (hex), O (ascii) – success ACK. or 0x46 (hex), F (ascii) – fail NAK.
Description	<p>This Command Opens or creates a file, depending on “Open Mode” parameter:</p> <p><i>Those commands are only for already created/existing files:</i></p> <ul style="list-style-type: none"> - Read Only: Specifies read access to the object, file data can only be read, not write. - Write Only: Specifies write access to the object, file data can only be write, not read. - Read+Write: Specifies read+write access to the object, file data can be write or read. <p><i>Those commands are only to create files:</i></p> <ul style="list-style-type: none"> - Create New: Creates a new file, command fails with “EXIST” (0x08) if the file exists. - Open Always: Opens the file if exists, if not, a new file is created. - Create Always: Creates a new file, if the file exists, it is truncated and overwritten.

	<p>After any “Open File” command succeeded, the file object is valid. The file object is used for subsequent read/write functions to identify the file. When close an open file object, use “Close File” function. If the modified file is not closed, the file data can be collapsed.</p> <p>Only one file can be opened at the same time, be sure to always close a working file before opening a new one.</p>
<p>Example (sent commands)</p>	<p>Example 1: Create a file named “0123.txt” that doesn’t exist, and then open it for write only access.</p> <p>-46,4F,04,30,31,32,33,2E,74,78,74,00- Creates new file “0123.txt”.</p> <p>-46,4F,02,30,31,32,33,2E,74,78,74,00- Open file “0123.txt” for write only access.</p> <p>Example 2: Open a file named “ABC.txt” that exists, for read only access.</p> <p>-46,4F,01,41,42,43,2E,74,78,74,00- Open existing file “0123.txt” for read only access.</p> <p>Example 3: Overwrite a file named “0123.txt” that exists, and then open it for read+write access.</p> <p>-46,4F,08,30,31,32,33,2E,74,78,74,00- Overwrite file “0123.txt”.</p> <p>-46,4F,03,30,31,32,33,2E,74,78,74,00- Open file “0123.txt” for read+write access.</p> <p>All data is in hex.</p>

2.5.2 Close File – 46hex - ‘F’ ascii + 43hex – ‘C’ ascii

Commands (host)	2 bytes
	1.- 0x46 (hex), F (ascii). 2.- 0x43 (hex), C (ascii).
Responses (device)	1 byte File ACK + 1 byte Command ACK
	1.- Refer to “File ACK List”. 2.- 0x4F (hex), O (ascii) – success ACK. or 0x46 (hex), F (ascii) – fail NAK.
Description	<p>The “Close File” command closes an open file object. If any data has been written to the file, the cached information of the file is written back to the disk. After the command succeeded, the file object is no longer valid and it can be discarded. If the modified file is not closed, the file data can be collapsed.</p> <p>If no file is opened during a “Close File” command, this will fail with NAK.</p> <p>Be sure to always close a working file before opening a new one.</p>
Example (sent commands)	<p>Example 1: Close any previously opened file.</p> <p>-46,43- Closes any open file.</p> <p>All data is in hex.</p>

2.5.3 Sync File – 46hex - ‘F’ ascii + 53hex – ‘S’ ascii

Commands (host)	2 bytes
	1.- 0x46 (hex), F (ascii). 2.- 0x53 (hex), S (ascii).
Responses (device)	1 byte File ACK + 1 byte Command ACK
	1.- Refer to “File ACK List”. 2.- 0x4F (hex), O (ascii) – success ACK. or 0x46 (hex), F (ascii) – fail NAK.
Description	<p>The “Sync File” command performs the same process as “Close File” command but the file is left opened and can continue Read/Write/set Pointer operations to the file. This is suitable for applications that open files for a long time in write mode, such as data logger.</p> <p>Performing “Sync File” of periodic or immediately after file write can minimize the risk of data loss due to a sudden blackout or an unintentional disk removal. However “Sync File” command immediately before “Close File” has no advantage because “Close File” performs “Sync File” in it. In other words, the difference between those functions is that the file object is invalidated or not.</p> <p>“Sync File” could also be seen as the “save disk icon” on any program, this let you save data but let program/file open.</p> <p>If no file is opened during a “Sync File” command, this will fail with NAK.</p> <p>Be sure to always close a working file before opening a new one.</p>
Example (sent commands)	<p>Example 1: Sync any previously opened file.</p> <p>-46,53– Sync open file.</p> <p>All data is in hex.</p>

2.5.4 Set Pointer – 46hex - 'F' ascii + 50hex – 'P' ascii

Commands (host)	6 bytes 1.- 0x46 (hex), F (ascii). 2.- 0x50 (hex), P (ascii). 3.- Position high byte. 4.- Position medium high byte. 5.- Position medium low byte. 6.- Position low byte. <i>*Note that the Position parameter is an Unsigned Long data type (4 bytes).</i>
Responses (device)	1 byte File ACK + 1 byte Command ACK 1.- Refer to "File ACK List". 2.- 0x4F (hex), O (ascii) – success ACK. or 0x46 (hex), F (ascii) – fail NAK.
Description	<p>The "Set Pointer" command moves the file read/write pointer of an open file object. It can also be used to increase the file size (cluster pre-allocation).</p> <p>The pointer is set/moved to the given number parameter from file origin to top.</p> <p>When an offset above the file size is specified in write mode, the file size is increased to the offset and the data in the expanded area is undefined. This is suitable to create a large file quickly, for fast write operation.</p> <p>Each time a "Read File" or "Write File" operation is performed, the file pointer advances the read/write bytes.</p> <p>If no file is opened during a "Set Pointer" command, this will fail with NAK.</p>
Example (sent commands)	-46,50,00,00,00,0A – Move file pointer to 10(dec) position (origin to top). -46,50,00,04,E6,7C – Move file pointer to 321148(dec) position (origin to top). All data is in hex.

2.5.5 Read File – 46hex - 'F' ascii + 52hex – 'R' ascii

Commands (host)	4 bytes 1.- 0x46 (hex), F (ascii). 2.- 0x52 (hex), R (ascii). 3.- Bytes to Read (High byte). 4.- Bytes to Read (Low byte).
Responses (device)	N Data bytes + 2 bytes (Successfully read) + 1 File ACK + 1 Command ACK 1 up to N.- File Data bytes.(<i>N= Bytes to Read</i>) N+1.- Successfully Read bytes (High byte). N+2.- Successfully Read bytes (Low byte). N+3.- Refer to “File ACK List”. N+4.- 0x4F (hex), O (ascii) – success ACK. or 0x46 (hex), F (ascii) – fail NAK.
Description	<p>The “Read File” command reads data from a previously opened file (“Open File”), set with “Read Only” or “Read+Write” mode.</p> <p>The file pointer of the file object increases in number of bytes read. After the command succeeded, Successfully Read bytes should be checked to detect the end of file. In case of Successfully Read bytes < Bytes to Read, it means the read/write pointer reached end of the file during read operation.</p> <p>This command always will try to read the Bytes to Read parameter, however note that even Successfully Read bytes < Bytes to Read, this command will always return the requested Byte to Read bytes, if Successfully Read bytes < Bytes to Read then the SmartGPU will complete/fill requested data bytes with 0x00(hex). Successfully Read bytes parameter will be the number of returned valid read bytes, the rest will be just 0x00(hex).</p> <p>Always after calling this command, the file pointer will increase the number of Successfully Read bytes from the last pointer position.</p> <p>If no file is opened during a “Read File” command, this will fail with NAK.</p>

Example (sent commands)

-46,52,00,0A– Read 10(dec) bytes from the current file pointer position (pointer will increase 10 positions after this command).

-46,52,13,88– Read 5000(dec) bytes from the current file pointer position (pointer will increase 5000 positions after this command).

-46,52,FF,FF– Read 65535(dec) bytes from the current file pointer position (pointer will increase 65535 positions after this command).

All data is in hex.

2.5.6 Write File – 46hex - ‘F’ ascii + 57hex – ‘W’ ascii

Commands (host)	4 bytes + N Data bytes 1.- 0x46 (hex), F (ascii). 2.- 0x57 (hex), W (ascii). 3.- Bytes to Write (High byte). 4.- Bytes to Write (Low byte). 5 up to N.- File Data bytes. (<i>N= Bytes to Write</i>). <i>*Max Bytes to Write parameter is 1900 (dec)</i>
Responses (device)	2 bytes (Successfully Write) + 1 File ACK + 1 Command ACK 1.- Successfully Written bytes (High byte). 2.- Successfully Written bytes (Low byte). 3.- Refer to “File ACK List”. 4.- 0x4F (hex), O (ascii) – success ACK. or 0x46 (hex), F (ascii) – fail NAK.
Description	<p>The “Write File” command writes data to a previously opened file (“Open File”), set with “Write Only” or “Read+Write” mode.</p> <p>The file pointer of the file object increases in number of written bytes. After the command succeeded, Successfully Written bytes should be checked to detect disk full. In case of Successfully Written bytes < Bytes to Write, it means the volume get full during write operation.</p> <p>Be sure to perform a “Sync File” or “Close File” command after each write cycle to save data and avoid data corruption.</p> <p>If no file is opened during a “Write File” command, this will fail with NAK.</p>
Example (sent commands)	-46,57,00,0A,(data to write)– Write 10(dec) bytes from the current file pointer position (pointer will increase 10 positions after this command). -46,57,07,6C,(data to write)– Write 1900(dec) (MAX) bytes from the current file pointer position (pointer will increase 1900 positions after this command). All data is in hex.

2.5.7 List Files –48hex – ‘H’ ascii

Commands (host)	1 byte
	1.- 0x48 (hex), H (ascii)
Responses (device)	FileName1, FileName2,...FileNameN + 1byte(NULL) + 1 byte Command ACK.
	1 up to N.- File Names separated by commas. N+1.- 0x00(hex) Null character, means end. N+2.- 0x4F (hex), O (ascii) – success ACK. or 0x46 (hex), F (ascii) – fail NAK.
Description	<p>The “List Files” command reads all files names from the micro SD and returns them one by one separated by a comma ‘,’ 0x2C(hex) character, and ends the list with a 0x00(hex) character. After this end 0x00 character an ACK or NAK is received.</p> <p>This is one of the simplest yet effective commands available on the Smart GPU FAT Data Management functions.</p> <p>This command could be called always and anytime.</p>

2.6 Touch Commands

The touch controller on the SMART GPU board, it's an accurate 10 bit ADC reader, perfectly debugged to avoid unwanted touch points.

This touch controller manages a resistive touch screen that is capable of handling finger touch and stylus pen touch, however for a precision drawing it's recommended the use of stylus.

The touch screen also contain 5 general purpose Icons drawn on the bottom corner, any touch on those Icons are processed on the touch controller and sent by the SMART GPU as touch on Icon, instead of just another coordinate touch.

Never use the touch with sharp objects or pens that are not designed for touch screens.

Briefly Summary of Commands in this section:

- Get Touch – 47hex 'G'
- Calibrate Touch – 48hex 'H'

2.6.1 Get Touch – 47hex - G ascii

Commands (host)	1 byte
	1.- 0x47 (hex), G (ascii).
Responses (device)	5 bytes
	1-4.- The touch coordinates or the Icon Name. + 5.- 0x4F (hex), O (ascii) – success ACK. or 0x46 (hex), F (ascii) – fail NAK.
Description	<p>Command needed to get the currently touch point on the screen, this command always read the touch screen and responds immediately to call, even if no touch is present, unlike other devices that wait until touch is pressed, SMART GPU responses immediately in order to release the main processor and avoid delay times.</p> <p>Possible responses:</p> <p>If no touch:</p> <p>'N' 'O' 'N' 'E' – 0x4E,0x4F,0x4E,0x45</p> <p>If touch on screen area:</p> <p>1.- X coord high byte. 2.- X coord low byte. 3.- Y coord high byte. 4.- Y coord low byte.</p> <p>If touch on Icons area:</p> <p>Touch on Home Icon: 'H' 'O' 'M' 'E' – 0x48,0x4F,0x4D,0x45</p> <p>Touch on Book Icon: 'B' 'O' 'O' 'K' – 0x42,0x4F,0x4F,0x4B</p> <p>Touch on Message Icon: 'M' 'E' 'S' 'G' – 0x4D,0x45,0x53,0x47</p> <p>Touch on Camera Icon: 'C' 'A' 'M' 'E' – 0x43,0x41,0x4D,0x45</p>

	<p>Touch on PC Monitor Icon: 'P' 'C' 'M' 'O' – 0x50,0x43,0x4D,0x4F</p> <p>Usually the Host sends this command until a response different than 'N' 'O' 'N' 'E' is received. Or also the host could wait while 'N' 'O' 'N' 'E' is received, that is until a touch is made.</p>
<p>Example (sent commands)</p>	<p>-47- Get touch. Received data: -0x01,0x15,0x00,0x56,0x4F- A touch at X:277(dec), Y:86(dec).</p> <p>-47- Get touch. Received data: -0x43,0x41,0x4D,0x45,0x4F- A touch on CAMERa Icon.</p> <p>-47- Get touch. Received data: -0x00,0xF5,0x00,0x24,0x4F- A touch at X:245(dec), Y:36(dec).</p> <p>-47- Get touch. Received data: -0x48,0x4F,0x4D,0x45,0x4F- A touch on HOME Icon.</p> <p>All data is in hex. Note: Maximum X or Y acceptable size values depend on display orientation.</p>

2.6.2 Calibrate Touch – 48hex - H ascii (ADVANCED USERS).

Commands (host)	11 bytes
	1.- 0x48 (hex), H (ascii). 2.- 0x55 (hex), U (ascii). 3.- XSubVal high byte. 4.- XSubVal low byte. 5.- YSubVal high byte. 6.- YSubVal low byte. 7.- XDivVal high byte. 8.- XDivVal low byte. 9.- YDivVal high byte. 10.- YDivVal low byte. 11.- Save to eeprom: 53(hex) 'S'. Save to ram: any other different byte.
Responses (device)	1 byte
	1.- 0x4F (hex), O (ascii) – success ACK. or 1.- 0x46 (hex), F (ascii) – fail NAK.
Description	<p>Command needed to set the touch calibration values, this command is recommended for advanced users and it's designed to work with the SMART GPU PC interface. However an advanced user could send those calibration values without the PC interface.</p> <p>After power up or reset, SMART GPU takes the eeprom calibration values and it loads them to ram.</p> <p>Factory default eeprom values:</p> <p>XSubVal : 0x017F(hex). YSubVal: 0x009C(hex). XDivVal: 0x0241(hex). YDivVal: 0x02EA(hex).</p>
Example (sent commands)	<div> -48,55,01,7F,00,9C,02,41,02,EA,00- values and save to ram only. Set </div> <div> -48,55,01,7F,00,9C,02,41,02,EA,53- values and save to eeprom. Set </div> <p>All data is in hex.</p>

2.7 Memory Read and Digital Out Commands

The SMART GPU contains a 1,843,200 bits internal memory, that stores the current display image on the screen, the command Memory Read, allows the host to be able to read this internal memory in order to save to an external file for example.

The colour convention for this memory read command is RGB888, instead of the RGB565, in order to avoid quality and colour depth loss, also the images that are read from the SD card are in RGB888 convention.

The Digital Out Commands, allows to control the 2 Digital Out pins available on the SMART GPU board.

Briefly Summary of Commands in this section:

- Memory Read – 4Dhex 'M'
- Digital Out Pin – 44hex 'D'

The colour responses of the memory read command consist of 24bits (3 bytes) RGB888:

R7R6R5R4R3R2R1R0 G7G6G5G4G3G2G1G0 B7B6B5B4B3B2B1B0

That is:


8bits for red, 8 bits for green, 8bits for blue.

High byte colour: R7R6R5R4R3R2R1R0

Medium byte colour: G7G6G5G4G3G2G1G0

Low byte colour: B7B6B5B4B3B2B1B0

2.7.1 Memory Read – 4Dhex - M ascii

Commands (host)	9 bytes
	1.- 0x4D (hex), M (ascii). 2.- X1 high byte. 3.- X1 low byte. 4.- Y1 high byte. 5.- Y1 low byte. 6.- X2 high byte. 7.- X2 low byte. 8.- Y2 high byte. 9.- Y2 low byte.
Responses (device)	pixel number x3
	1-N- pixels defined by the X1(16bit),Y1(16bit), and X2(16bit),Y2(16bit) window. N+1.- 0x4F (hex), O (ascii) – success ACK. or 0x46 (hex), F (ascii) – fail NAK. <i>Remember: each pixel is composed by three bytes following the RGB888 convention.</i>
Description	This command reads the internal memory of the SMART GPU that is currently displayed, defined by the given points(top left corner): X1(16bit),Y1(16bit) and (bottom right corner)X2(16bit),Y2(16bit).
Example (sent commands)	<p>The next examples are considered to be executed when the next image is being displayed on the SMART GPU screen.</p> 

-4D,00,00,00,00,00,A0,00,78- Reads the contents of the SMART GPU memory from the top left corner: X1:00(dec),Y1:00(dec) and bottom right corner: X2:160(dec), Y2:120(dec), and returns the next image of 160x120 pixels, sent as RGB each pixel.



-4D,00,50,00,3C,00,F0,00,B4- Reads the contents of the SMART GPU memory from the top left corner: X1:80(dec),Y1:60(dec) and bottom right corner: X2:240(dec), Y2:180(dec), and returns the next image of 160x120 pixels, sent as RGB each pixel.



-4D,00,A0,00,00,00,F0,00,78- Reads the contents of the SMART GPU memory from the top left corner: X1:160(dec),Y1:00(dec) and bottom right corner: X2:240(dec), Y2:120(dec), and returns the next image of 80x120 pixels, sent as RGB each pixel.

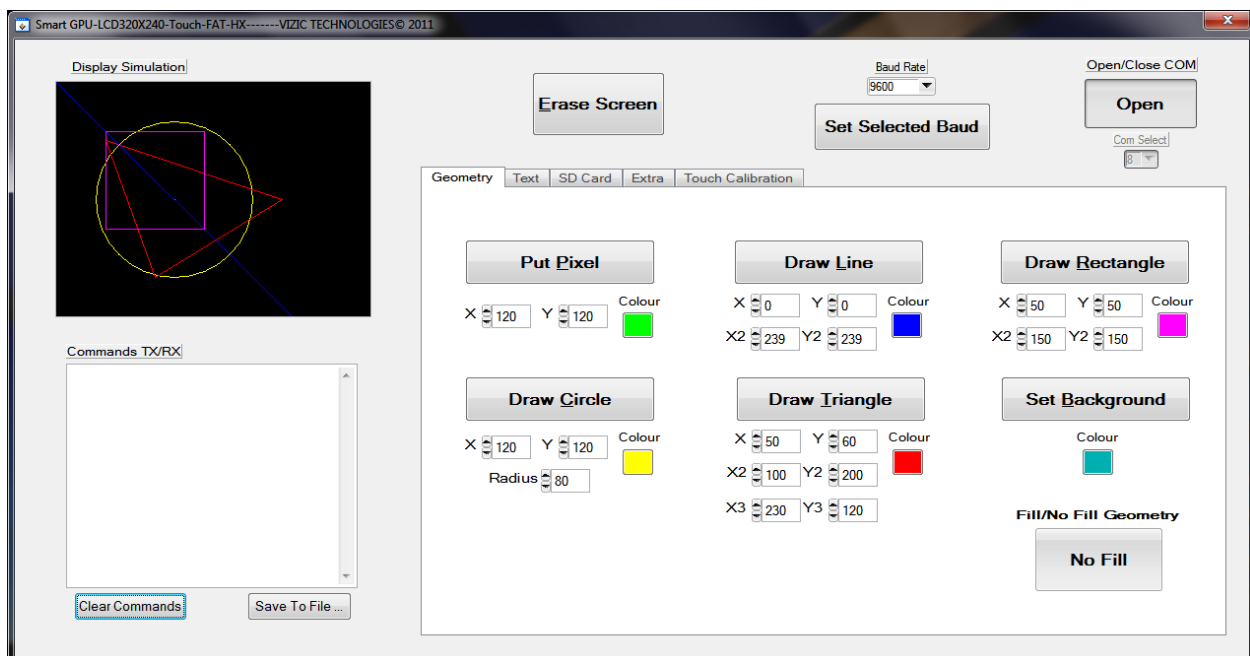


All data is in hex. **NOTE:** Maximum X1, Y1, X2, Y2 acceptable size values depend on display orientation.

3 Development software tools

In order to make easier the learning about how to communicate with the SMART GPU, free software could be downloaded and used in any PC. This software simulates most of the functions of the SMART GPU by connecting the USB-UART SX Bridge to SMART GPU to enable real live graphics processing.

This software greatly reduces the time of learning the commands, and helps the user to understand how commands are created.



For detailed information about this software and how to use it, please refer to the “SMARTGPU-PCsimulation.pdf” sheet that could be downloaded in the web site.

For detailed information about the USB-UART SX bridge, please visit our web site.

VIZIC TECHNOLOGIES. COPYRIGHT 2012.

THE DATASHEETS AND SOFTWARE ARE PROVIDED "AS IS." VIZIC EXPRESSLY DISCLAIM ANY WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT.

IN NO EVENT SHALL VIZIC BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, HARM TO YOUR EQUIPMENT, COST OF PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY OR SERVICES, ANY CLAIMS BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENCE THEREOF), ANY CLAIMS FOR INDEMNITY OR CONTRIBUTION, OR OTHER SIMILAR COSTS.

Proprietary Information:

The information contained in this document is the property of Vizic Technologies and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission.

Vizic Tech endeavors to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission. The development tools of Vizic products and services are continuous and published information may not be up to date. It is important to check the current position with Vizic Technologies at the web site.

All trademarks belong to their respective owners and are recognized and acknowledged.

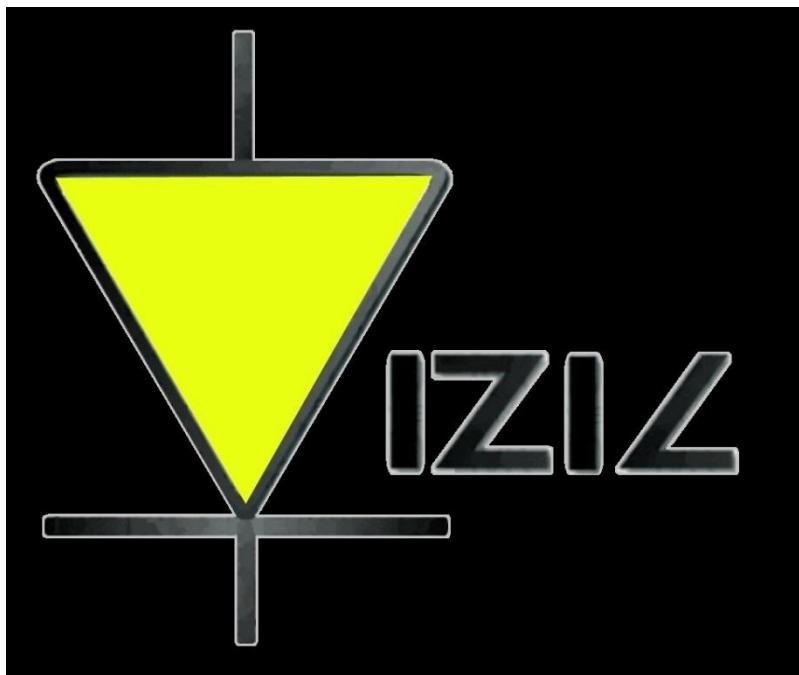
Disclaimer of Warranties & Limitation of Liability:

Vizic Technologies makes no warranty, either expresses or implied with respect to any product, and specifically disclaims all other warranties, including, without limitation, warranties for merchantability, non-infringement and fitness for any particular purpose.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

In no event shall Vizic be liable to the buyer or to any third party for any indirect, incidental, special, consequential, punitive or exemplary damages (including without limitation lost profits, lost savings, or loss of business opportunity) arising out of or relating to any product or service provided or to be provided by Vizic Tech, or the use or inability to use the same, even if Vizic has been advised of the possibility of such damages.

Use of Vizic' devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Vizic Technologies from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Vizic Technologies intellectual property rights.



[www.VIZICTECHNOLOGIES.COM](http://www.vizictechnologies.com)