



HAIDAR TECHNOLOGY, LLC.
The Next Generation Of Intelligent Embedded GUI Systems

WWW.haidartecchnology.com

(614) 389-3022

Sales@haidartecchnology.com

SLCD-320240M21-5.7xx
Serial QVGA (320X240) LCD Panel With Touch Screen
And Integrated GUI Objects

Revision 1.00

Issue Date: 10/22/2010

© Copyright Haidar Technology 2007-2010

Important Notice:

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

1- Overview:

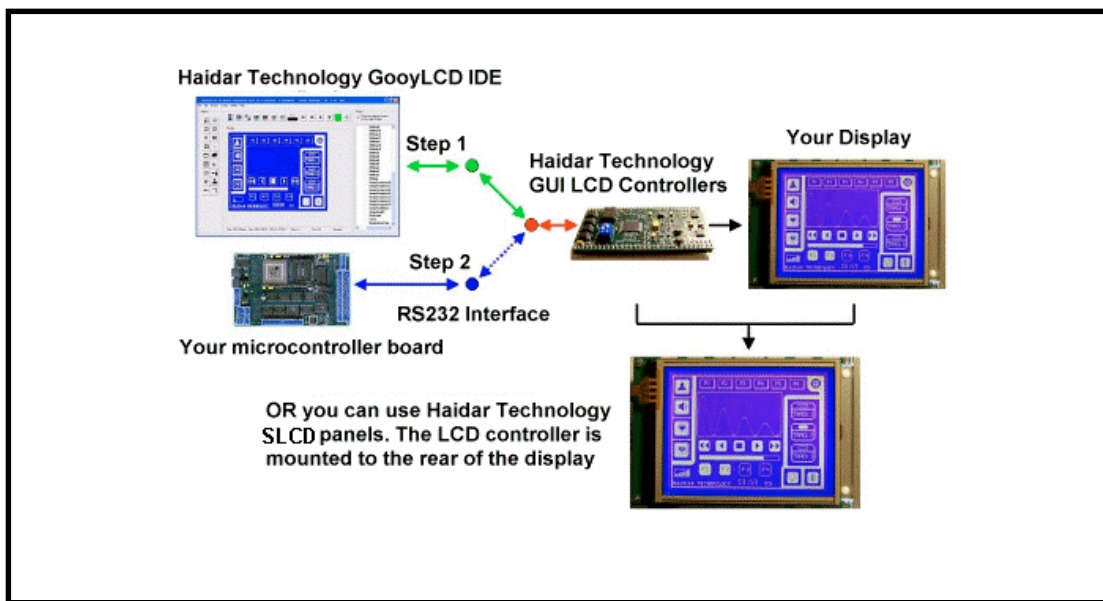
The sLCD series of displays are the first displays with integrated Graphical User Interface (GUI) capability. In addition to a variety of integrated fonts that can be used with pixel accuracy, they offer a whole range of sophisticated graphics functions.

The displays are ready for operation immediately with an operating voltage of 5V. They are controlled via RS232 interface. The displays are “programmed” by means of high-level language-type graphics commands. There is no longer any need for the time-consuming programming of character sets and graphics routines. The ease of use of this display with its touch panel dramatically reduces development times.

“GooyLCD IDE“ software allows you to edit bitmaps, images, fonts and design a graphics-rich GUI application from a desktop computer without writing a single line of code! With GooyLCD IDE you will be able to visually design your GUI program screens (up to 24 screens) using a rich selection of GUI objects. Each screen is compiled by the IDE into a binary file, which will be saved in the module EEPROM.

By using the Object Run Time Simulation utility, you will be able to simulate and test the objects run time properties before even your hardware and code are ready. For example, you can run an animation on the target display before your host controller code is ready.

If the layout or some bitmaps need to be redesigned just disconnects the target hardware from the controlling processor (host) and moves it to a serial port of a PC. Program the new code and you have the new user interface in place! Pure visual changes do not require modifications to the host controller code!



Step 1: Create your GUI (up to 24 screens) using Haidar Technology “GooyLCD IDE” software. Edit your own bitmaps and fonts, load images and simulate your GUI from your PC screen. The IDE compiles the GUI into a binary file and saves it into the LCD controller EEPROM. It is simple, fun and fast!

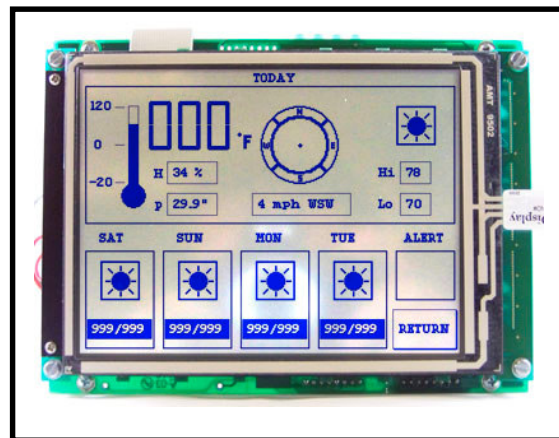
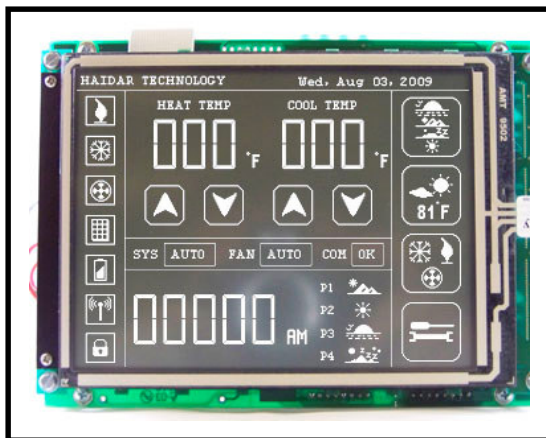
Step 2: Your controller sends high level serial (RS232) commands to control the GUI you just created in step 1. The rich set of powerful object and terminal commands makes controlling the objects easy and simple. No need for RTOS or special library!

Ordering information:

Product #	Display	TS	BL
sLCD-320240M21-5.7WL-T	NewHaven (FSTN+) NHD-320240WG-BxSFH-VZ#	Yes	White LED
sLCD-320240M21-5.7BL-T	NewHaven (STN-Blue) NHD-320240WG-BxTMI-VZ#	Yes	White LED

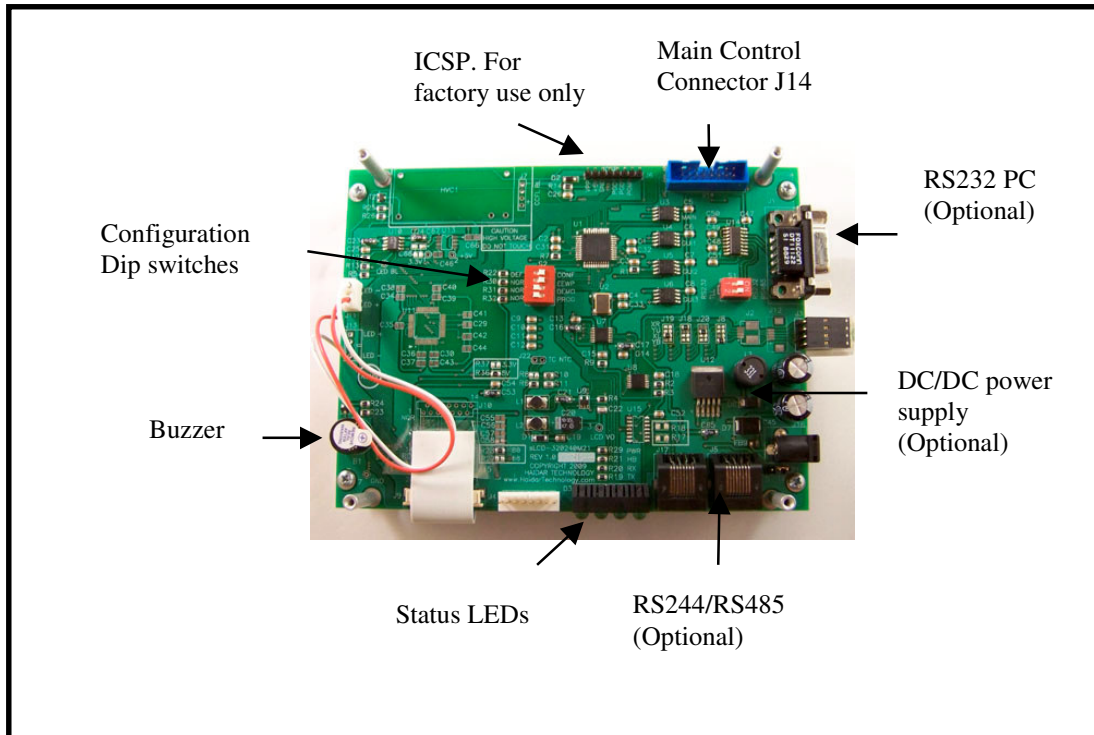
2- Features:

- 5.7" QVGA (320X240) graphic monochrome LCD with LED back light and 4-wire resistive touch screen
- Available in White/Black or Blue/White background
- RS232 TTL interface
- RS422/485 interface on board (Optional)
- RS232 PC interface on board (Optional)
- DC/DC power supply on board (Optional)
- Operate from a single power supply 5V
- Powerful Resistive Touch Screen controller on board
- DC/DC negative/positive LCD bias supply on board
- Low Drop power regulator for the LED back light
- CCFL DC/AC inverter for LCD with CCFL back light (Optional)
- 512Kbytes of EEPROM for fonts/bitmap and GUI program (Up to 24 screens) storage
- Graphical Button Object (One bitmap for the Pressed State and other for Unpressed state)
- Vector/Point Chart Type
- Reverse/Normal Display Mode
- Single Object Command to fill the Chart Object (Array Command)
- Huge Digits (64X128) for the NumberBox Object
- Image Memory Location can be used for the extra 64 32X32 bitmaps and the Huge NumberBox object digit bitmaps
- TextBox and Label can be set to act as TextField object
- Software controlled contrast
- Software controlled brightness
- Software controlled backlight ON/OFF
- Programming mode allows for direct access to the module EEPROM
- 3 analog inputs and horizontal scanning signal for chart update (oscilloscope mode)
- Output signals for Alarm, Buzzer, Heartbeat and communication status
- Two modes of operation: Terminal and Object mode
- 6 proportional fonts
- 8 Precoded Text Objects (Label, Frame, TextBox, TextField, Button, RadioButton, CheckBox)
- 4 Precoded Primitive Objects (Line, Circle, Ellipse, Rectangle)
- 3 Precoded Picture Objects (Bitmap, ImageSeq, Animation)
- 5 Precoded HMI Objects (Chart, BarGraph, NumberBox, LinearGage, Slider)
- Touch screen calibration algorithm

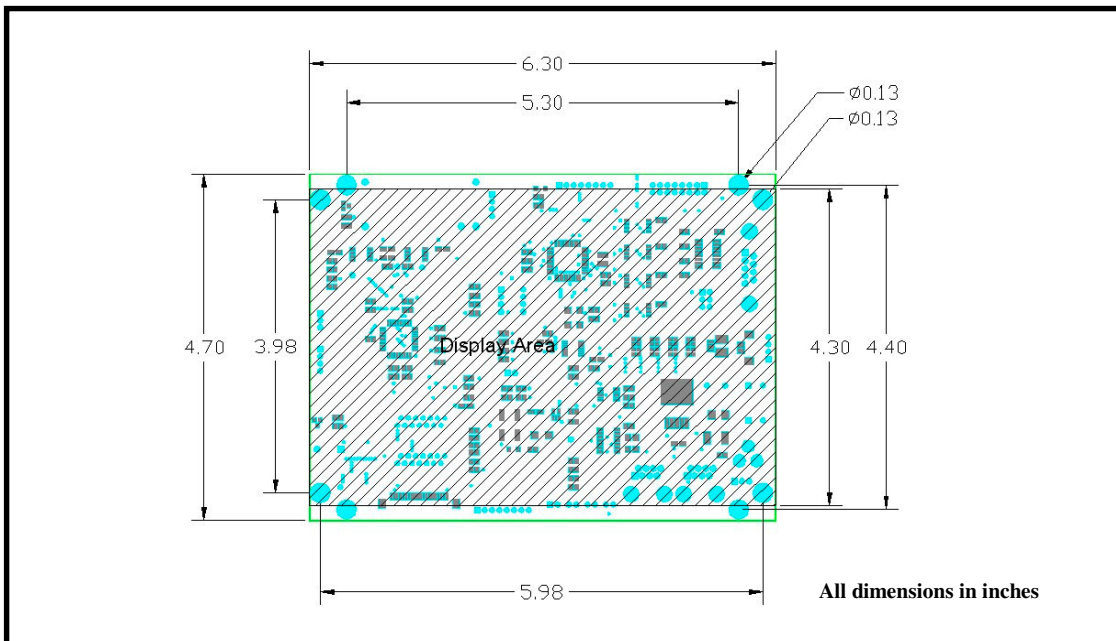


3- Board Picture:

Shown with the optional RS422/RS485, RS232 Driver and DC/DC power supply



4- Board Dimensions:



5- Connectors Description:

- **Main Control J14:** 16-pin 0.1" IDC connector

Pin Number	Pin Name	Pin Type	Pin Description
1	+5V	PWR	+5V DC In (500 MA)
2	+12V	PWR	+12V DC for CCFL HVC (Only if the LCD back light is CCFL)
3	GND	PWR	Ground reference
4	GND	PWR	Ground reference
5	RS232-Rx	TTL IN	RS232 Receiver
6	RS232-Tx	TTL OUT	RS232 Transmitter
7	RS422-DE	TTL OUT	RS422 Data Enable
8	AIN-CH0	Analog IN	CH0 Analog input (0 – 5V) for chart update
19	AIN-CH1	Analog IN	CH1 Analog input (0 – 5V) for chart update
10	AIN-CH2	Analog IN	CH2 Analog input (0 – 5V) for chart update
11	HSS	TTL IN	Horizontal Scanning Signal for chart update See the Chart object for more details
12	Alarm-LED	TTL OUT	This output can be used as general visual alarm controlled by software. LED with limiting resistor can be connected to this pin.
13	Buzzer	TTL OUT	This output can be used as general audible alarm controlled by software. A buzzer with proper current sink can be connected to this pin (This pin is already connected to the buzzer on board)
14	BL-ON/OFF	TTL OUT	This TTL Output indicates the status of the LCD back light. 1 => ON 0 => OFF It can be used to turn external light (front panel lighting) ON or OFF
15	BL-LED +	PWR out	This pin can be used to drive external light (LED).
16	BL-LED -	GND	

- **Auxiliary Control J4:** 8-pins 0.1" connector (Molex # 0022112082)

Pin Name	Pin Number	Pin Type	Description
VDD	1	PWR	Power Input (+12V). Only if the optional power supply is populated
GND	2	PWR	Ground reference (0V)
RS232_RX	3	RS232 PC Level ($\pm 12V$)	RS232 Receiver. Available when the RS232 driver chip (U14) is populated
RS232_TX	4	RS232 PC Level ($\pm 12V$)	RS232 Transmitter. Available when the RS232 driver chip (U14) is populated
RS422_T+	5	RS422/485 PC Level ($\pm 12V$)	RS422/485 Transmitter (+). Available when the RS422 driver chip (U15) is populated
RS422_T-	6	RS422/485 PC Level ($\pm 12V$)	RS422/485 Transmitter (-). Available when the RS422 driver chip (U15) is populated
RS422_R+	7	RS422/485 PC Level ($\pm 12V$)	RS422/485 Receiver (+). Available when the RS422 driver chip (U15) is populated
RS422_R-	8	RS422/485 PC Level ($\pm 12V$)	RS422/485 Receiver (-). Available when the RS422 driver chip (U15) is populated

Auxiliary Control J5 and J17 (RS422/485): RG45 connectors

Pin Name	Pin Number J5 and J17	Pin Type	Description
RS422_T+	1	RS422/485 PC Level ($\pm 12V$)	RS422/485 Transmitter (+). Available when the RS422 driver chip (U15) is populated
RS422_T-	2	RS422/485 PC Level ($\pm 12V$)	RS422/485 Transmitter (-). Available when the RS422 driver chip (U15) is populated
RS422_R-	3	RS422/485 PC Level ($\pm 12V$)	RS422/485 Receiver (-). Available when the RS422 driver chip (U15) is populated
VDD	4	PWR	Power Input (+12V). Only if the optional power supply is populated
VDD	5	PWR	Power Input (+12V). Only if the optional power supply is populated
RS422_R+	6	RS422/485 PC Level ($\pm 12V$)	RS422/485 Receiver (+). Available when the RS422 driver chip (U15) is populated
GND	7	PWR	Ground reference (0V)
GND	8	PWR	Ground reference (0V)

6- Electrical Characteristics:

Recommended Operation Conditions

Symbol	Parameter	Condition	Min	Typ	Max	Unit
VDD	Supply voltage	GND = 0 V	4.5	5.0	5.5	V
VAIN	Analog input voltage	GND = 0 V	0	5.0	5.5	V
VLED	Supply voltage for LED backlight	GND = 0 V	1.6		3.5	V
IL	Supply current VDD = +5V	LED BL ON	-	180	250	MA
		LED BL OFF	-	50	-	
VEE	Supply negative voltage		-20		-12	V
VOH	High level output voltage	GND = 0 V TTL level	VDD- 0.4			V
VOL	Low level output voltage	GND = 0 V TTL level			0.8	V
VIH	High level input voltage	GND = 0 V TTL level	2.0			V
VIL	Low level input voltage	GND = 0 V TTL level			0.8	V

7- Modes of Operation:

7-1- Terminal Mode:

The terminal mode commands can be used at any time by the host controller to configure or control the module/LCD. In this mode, the host controller is responsible for creating the graphical user interface and response to the user touch using the provided commands. The terminal mode commands are divided into 3 groups:

- Configuration and Control
- Graphics and Text
- Touch screen

The terminal commands will be discussed in details latter in this manual.

GooyLCD IDE provides 7 utilities, which includes Terminal, Contrast, Brightness, Alarm, Buzzer, and Touch screen, Configure Module to test the function of terminal commands.

7-2- Object Mode:

In this mode of operation, the user will be able to design a graphical user interface application visually and screen by screen (up to 24 screen) using GooyLCD Mono IDE software. The GUI application is created using the included set of powerful objects and then compiling the application screens into binary files that will be saved into the controller EEPROM. The host controller will use the object commands to alter or redraw the objects through their run time properties.

The object commands will be discussed in details later in this manual.

8- EEPROM:

The module has 512Kbytes of external EEPROM for Fonts, bitmaps, and images and GUI program storage. The external EEPROM is divided into four chips each one has 128Kbytes:

- Fonts/Bitmaps EEPROM : store fonts, bitmaps and images
- GUI1 EEPROM : store GUI Program screen 0 to 7
- GUI2 EEPROM: store GUI program screen 8 to 15
- GUI3 EEPROM: store GUI program screen 16 to 23

To prevent an accidental write to the EEPROM during normal operation, set the DIPswitch to WP.

9- Programming Mode:

The programming mode is used to program the blank controllers without the need for LCD. **In this mode, the controller will automatically configured to 115.2kb/sec baud rate and [16] device ID (Comm address), the firmware will skip the LCD initialization routine.** The programming mode provides a direct access to the module EEPROM for reading or writing operation.

After you finish from your GUI program including fonts, bitmaps and images, you can make an exact copy of your program by reading the module EEPROM and save it to HEX files. Those files will be the your master files to program the other blank module by writing them to their EEPROM.

To enter the programming mode:

- 1- Turn off the module and the display
- 2- Disconnect the display
- 3- Set the DIP switch to PM
- 4- Turn On the module

10- Display Interface:

The controller board is capable of driving/controlling any monochrome graphic display with the following specifications:

- 1- 320X240 resolution (different resolution is possible upon request)
- 2- With or without built in graphic controller
- 3- LED or CCFL backlight
- 4- 3.3V or 5V supply voltage

11- Communication Interface:

A simple RS232 or RS422/485 serial interface is implemented to communicate with the host controller or the PC. Three pins are provided for this interface:

- TX: Transmitter (TTL Level)
- RX: Receiver (TTL Level)
- DE: Data Enable (TTL Level)

DE (active high) is used to implement RS422/485 serial communication and is used to turn the transmitter ON only during transmission.

RS232 settings are:

- 8 bit data
- 1 start bit
- 1 stop bit
- No parity
- No flow control

The baud rate is user defined (default is 19200 bit/sec) and it can be set to one of the followings values:

9600 bit/sec
19200 bit/sec
38400 bit/sec
57600 bit/sec
115200 bit/sec

sLCD-320240M21-5.7xx requires 5.0V DC. Exceeding the supply voltage over the typical value (5.0) will cause a permanent damage to the board and to the attached LCD and void your warranty.

Warning: RX and TX use a TTL level of 5.0V. Connecting them to standard (PC) RS232 with +/- 12V or other will damage the controller and void your warranty.

The communication protocol is a simple ACK/NAK protocol with the following format:

- Command format

Device ID
LB
CMD
Data Field
CS

Higher baud rate than 19.2 kb/sec can cause communication problems (not fully tested).
 19.2 kb/sec is fully tested and it is suitable for most applications.
 115.2 kb/sec is also fully tested and only can be used in the programming mode

- ACK format

Device ID
ACK Code
LB
Data Field
CS

- NAK format

Device ID
NAK Code
CS

Where:

- Device ID: is the module communication address (default is 0x10)
- LB: is the number of bytes to follow excluding the check sum (CS)
- CMD: is the command code
- CS: is the check sum and is the LS byte of the summation of all the bytes in the packet
- ACK code = 0x06
- NAK code = 0x15

You will receive a NAK if:

- Command is not in the list
- Wrong number of command arguments
- Invalid argument

12- Fonts:

GLC-Mxx can write text in six different user editable proportional fonts. Characters can be placed at any pixel in the view area of the display. Proportional fonts have more advantages over mono spaced fonts, they are easier to read and they save more space, which make them ideal for small, low resolution displays. Each font has 128 character set [ASCII codes (0x20 – 0x9F)]. ASCII codes from 0x00 to 0x1F must not be used.

Font name	Cell Size	Font Code	Number of Characters
Font1	8X8	1	128
Font2	8X8	2	128
Font3	8X16	3	128
Font4	8X16	4	128
Font5	16X24	5	128
Font6	16X24	6	128

13- Touch Screen:

GLC-Mxx modules use Texas Instruments model TSC2003, 4 wire resistive touch screen controller. It features a 12-bit accuracy; touch pressure measurement and hardware interrupt “PENIRQ”. A firmware algorithm “TSCControl” is implemented to measure and correct the touch data (X, Y, Z1, Z2 and TStatus) and calibrate the touch screen.

1. Modes of operation:

TSCControl is controlled through the “TSConfigByte” and can operate in one of these modes:

- Interrupt, touch data are sent on request (Mode 0): Every time the user touches the touch screen, an interrupt is generated at the falling edge of “PENIRQ”. Touch data are measured and saved in the touch data buffer to be read by the host controller using the command “Get Touch Data”
- Interrupt, touch data are sent on every interrupt (Mode 1): Every time the user touches the touch screen, an interrupt is generated at the falling edge of “PENIRQ”. Touch data are measured and sent to the host controller.
- Continuous, touch data are sent every 100 msec (Mode 2): In this mode, TSControl is continuously scanning the touch screen every 100 msec, touch data are measured and sent to the host controller every 100 msec. You can stop sending the touch data by setting the “TSConfigByte” to Mode 3
- Continuous, touch data are sent on request (Mode 3): This mode also is the “Object Touch Mode” and it is the default mode. Tscontrol is continuously scanning the touch screen every 100 msec, touch data are measured and saved in the touch data buffer to be read by the host controller using the command “Get Touch Data”.

2. Calibration:

Several sources of errors affect the X and Y coordinates produced by the touch screen. The most important sources of error are electrical noise, mechanical misalignments, non-linearity, and scaling factors. The calibration algorithm permits the elimination of the scaling factors and mechanical misalignments of the touch screen.

The challenge of the calibration algorithm is to translate the set of coordinates reported by the touch screen into a set of coordinates that accurately represent a point on the display.

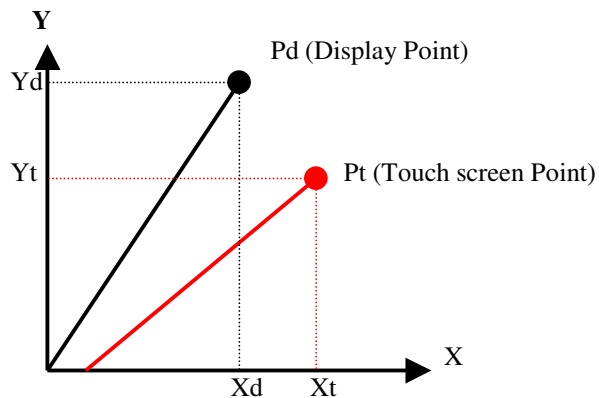
$$[X_d, Y_d] = f([X_t, Y_t])$$

Where:

X_d, Y_d : are the display coordinates

X_t, Y_t : are the coordinates reported by the touch screen

f : is calibration algorithm function



The above figure shows three factors of errors:

- Rotation of touch screen coordinates relative to the display coordinates
- Linear shift of coordinates
- A scaling factor

In this case, the calibration algorithm needs to calculate six calibration coefficients (3 point calibration algorithm). The corrected touch screen coordinates are calculated from following expressions:

$$\begin{aligned} X_d &= A(X_t) + E(Y_t) + B \\ Y_d &= C(X_t) + F(Y_t) + D \end{aligned}$$

But, if we assume the rotation error (tilting) is too small ($E = F = 0$), then the calibration algorithm needs only to calculate four calibration coefficients (2 points calibration algorithm) reducing the expressions to:

$$\begin{aligned} X_d &= A(X_t) + B \\ Y_d &= C(X_t) + D \end{aligned}$$

Where:

A, B, C, and D are the calibration coefficients or constants.

TSCControl uses a two points calibration algorithm to calibrate the touch screen. There is no need to calibrate the touch screen every time the device is powered on. The calibration constants are saved in the internal EEPROM and they will restore after power on.

The calibration procedure is as following:

After you send “CAL TS” command to the module, the screen will be cleared and a small circle will appear at the upper left corner of the display (P1). Touch this circle with a stylus and remove your hand. The circle will disappear and another circle will appear at the other opposite corner (P2). Again touch this circle with a stylus and then remove your hand. The circle will disappear and the touch screen calibration is done.



3. Touch Pressure:

TSC2003 provides a mean to measure the touch pressure by measuring the two components Z1 and Z2. Touch pressure can be used to differentiate between a finger touch or stylus touch and it can be calculated from the following formula:

$$\text{Touch Pressure} = R_{x\text{-plate}} * (X_t / 4096) * (Z_2 / Z_1 - 1)$$

For relative measurement of the touch pressure, you can assume $R_{x\text{-plate}} = 1$ and the relative touch pressure will be:

$$\text{Relative Touch Pressure} = (X_t / 4096) * (Z_2 / Z_1 - 1)$$

Where:

X_t : is the touch X coordinate (Uncorrected TSCConfigByte <R/C> = 0).

4. TSCConfigByte: Touch Screen Configuration Byte

7	6	5	4	3	2	1	0
M1	M0	R/C	Z	A	AV1	AV0	E

Bit0: Enable

0 = Touch screen controller is disable

1 = Touch screen controller is enable

Bit1,2: Touch Screen data average select

00 = No average

01 = 4 samples average

10 = 8 samples average

11 = 16 samples average

Bit3: Audible Touch (only available for Mode 0 and 1)

0 = Silence touch
1 = Audible touch

Bit4: Measure Z1 and Z2

0 = Do not measure Z1 & Z2
1 = Measure Z1 & Z2

Bit5: Send Raw or Corrected data

0 = Send Raw data
1 = Send Corrected Data

Bit6,7: Mode select

00 = Interrupt mode, data are sent on request (using "Get TS Data" command)
01 = Interrupt mode, data are sent on every touch
10 = Continuous mode, data are sent every 100 msec
11 = Continuous mode (Object Touch mode), data are sent on request

5. Tstatus: Touch Status

TStatus

7	6	5	4	3	2	1	0
-	-	-	DT	IT	OT	ST	CIP

Bit0: Touch screen calibration in progress

0 = Normal operation
1 = Calibration in progress

Bit1: Screen Touch

0 = Screen is not being touched
1 = Screen is being touched

Bit2: Object Touch

0 = Object is not being touched
1 = Object is being touched

Bit3: Increment Touch (ListBox increment bitmap)

0 = Increment ListBox bitmap is not being touched
1 = Increment ListBox bitmap is being touched

Bit4: Decrement Touch (ListBox decrement bitmap)

0 = Decrement ListBox bitmap is not being touched
1 = Decrement ListBox bitmap is being touched

Important Notes:

1. In order to calibrate the touch screen, the "TsConfigByte" must be set as following:

E = 1

AV1-AV0 = 2

A = 0

Z = 0

R/C = 1

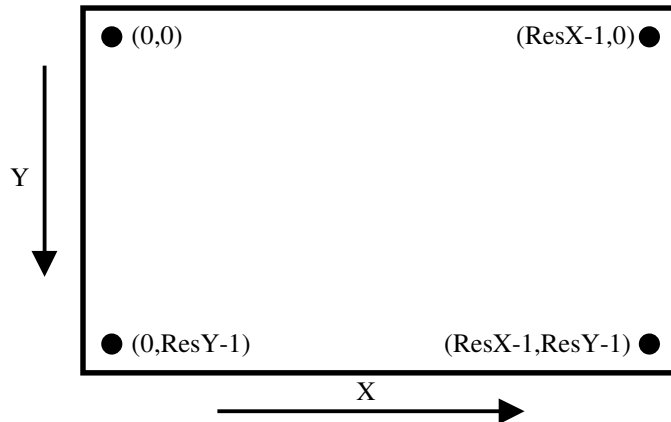
M1-M0 = 3 (Mode 3)

The above settings are the default settings upon power up.

2. In the current firmware, the touch screen data average is fixed to 8 samples average.

14- Screen Origin:

The screen origin (0,0) is the upper left corner of the display.



1- Terminal Commands:

- **RESET**

This command is used to RESET the module to its initial settings and it is equivalent to a hardware reset

CMD Code	0x01
CMD format	DVID,0x01,0x01,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

- **Set Configuration**

This command is used to write 16 configuration bytes to the module. A software or hardware reset is required after sending this command for the new configuration to take place.

CMD Code	0x05
CMD format	DVID,0x11,0x05, [16 configuration bytes B0 – B15], CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

Configuration bytes (The default settings are set when Dip Switch 1 is in the OFF position)

0	Device ID	Default = 16
1	ConfigByte	Default = 1
2	Baud Rate	Default = 19.2KB/Sec
3	0	0
4	LCDResX_H	1
5	LCDResX_L	28
6	LCDResY_H	0
7	LCDResY_L	240
8	0	-
9	0	-
10	0	-
11	APU-BL	Default = ON
12	APU-SS	Default = NO

13	0	-
14	0	-
15	0	-

LCDResX_H&LCDResX_L: LCD Horizontal Resolution

LCDResY_H&LCDResY_L: LCD Vertical Resolution

Note: LCD resolution is fixed to 320X240

ConfigByte: Controller Configuration Byte

ConfigByte Bit #	Description
0	TouchScreen Controller Enable/Disable 1: Enable (Default) 0: Disable
1	LCD Mode Select (Normal/Reverse) 1: Reverse 0: Normal (Default)
2	Use Image Memory Location 1: For the extra 64 32X32 bitmaps and the 23 huge digits bitmaps 0: For Images (Full screen bitmaps) (Default)
3	NA
4	NA
5	NA
6	NA
7	NA

BR-SEL: Baud Rate Select

BR-SEL	Baud Rate
0	9600 bit/sec
1	19200 bit/sec (Default)
2	38400 bit/sec
3	57600 bit/sec
4	115200 bit/sec

APU-BL: At Power Up, Backlight is ON or OFF

APU-BL	Backlight
0	OFF
1	ON (Default)

APU-SS: At Power Up, Display Splash Screen (Image ID = 0)

APU-SS	Splash Screen
0	NO (Default)
1	Yes

• Get Module Info

This command is used to get the module serial number, product number and firmware version.

CMD Code	0x02
CMD format	DVID,0x01,0x02,CS
ACK format	DVID,0x06,0x10,[16 bytes info B0-B15],CS
NAK format	DVID,0x15,CS

0	PN1
1	PN2
2	PN3
3	PN4
4	PN5

5	PN6
6	SN1
7	SN2
8	SN3
9	SN4
10	SN5
11	SN6
12	FV1
13	FV2
14	FV3
15	FV4

PN: Product Number (6 CHR)
 SN: Serial Number (6 CHR)
 FV: Firmware Version (4 CHR)

- **Set LCD Mode**

This command is used to turn the display ON or OFF and to change the display mode from Normal to Reverse.

CMD Code	0x06
CMD format	DVID,0x02,0x0D, Control Byte ,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

Control Byte

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	LCD ON/OFF

Bit0: LCD ON/OFF
 0 = LCD OFF
 1 = LCD ON

- **Adjust Contrast**

This command is used to adjust the display contrast level. A value from 0 to 255 can be used as a contrast level. This value will be saved in the digital POT EEPROM so it will survive power down or a reset.

CMD Code	0x15
CMD format	DVID,0x02,0x15, Contrast Level ,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

- **Adjust Brightness**

This command is only can be used for displays with LED backlight. It is used to adjust the brightness of the display by controlling the voltage drop across the backlight LED. A value from 0 to 255 can be used as brightness level. This value will be saved in the digital POT EEPROM so it will survive a power down or a reset.

CMD Code	0x16
CMD format	DVID,0x02,0x16, Brightness Level ,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

- **Backlight ON/OFF**

This command is used to turn the display backlight ON or OFF.
 Control Byte = 0 => Backlight is OFF

Control Byte = 1 => Backlight is ON

CMD Code	0x1C
CMD format	DVID,0x02,0x1C, Control Byte ,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

- **Set Alarm**

This command is used to set the visual and audible alarm.

CMD Code	0x17
CMD format	DVID,0x02,0x17, Alarm Mode ,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

Alarm Mode

7	6	5	4	3	2	1	0
Aud_alarm	x	x	x	x	x	M1	M0

Bit7: Audible Alarm

0 = No audible alarm (only visual alarm)

1 = audible and visual alarm

Bit1-0: Alarm Mode Select (M1-M0)

00 = Alarm is OFF (Both audible and visual)

01 = Alarm is ON Solid

10 = Alarm is ON Flashing Slow

11 = Alarm is ON Flashing Fast

- **Beep**

This command is used to activate (beep) the buzzer for certain number of msec indicated by Beep-Time.

A value from 0 to 255 msec can be used for the Beep-Time.

CMD Code	0x18
CMD format	DVID,0x02,0x18, Beep-Time ,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

- **Clear Screen**

This command is used to clear the entire screen text and graphics.

CMD Code	0x0B
CMD format	DVID,0x01,0x0B,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

- **Draw Pixel**

This command is used to place (draw) a pixel at (X,Y) coordinates.

CMD Code	0x0C
CMD format	DVID,0x11,0x0C,X_H,X_L,Y_H,Y_L,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,Visible,0x00,0x00 ,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

X: 0 to LCDResX - 1

Y: 0 to LCDResY – 1

Visible: 0 = The pixel is not visible (clear the pixel)

1 = The pixel is visible (draw the pixel)

- **Draw Line**

This command is used to draw a line from P1(x1,y1) to P2(x2,y2)

CMD Code	0x0D
CMD format	DVID,0x11,0x0D,X1_H,X1_L,Y1_H,Y1_L,X2_H,X2_L,Y2_H,Y2_L,0x00,0x00,0x,00,0x00,0x00,Visible,DrawStyle,0x00,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

X1: 0 to LCDResX – 1

Y1: 0 to LCDResY – 1

X2: 0 to LCDResX – 1

Y2: 0 to LCDResY – 1

Y1 must be equal or greater than Y2

Visible: 0 = The Line is not visible (clear the line)

1 = The Line is visible (draw the line)

DrawStyle: see Draw style table

- **Draw Circle**

This command is used to draw a circle with radius (Ra) and centered at (X0,Y0).

CMD Code	0x0F
CMD format	DVID,0x11,0x0F,X0_H,X0_L,Y0_H,Y0_L,Ra,0x00,0x00,0x00,0x00,0x00,0x00,0x00,Visible,0x00,FillStyle ,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

X0: 0 to LCDResX –1

Y0: 0 to LCDResY – 1

Ra: 0 to 255 (circle radius must be less than LCDResX –1 and LCDResY – 1)

Visible: 0 = The Circle is not visible (clear the circle)

1 = The Circle is visible (draw the Circle)

FillStyle: see fill style table (only FillStyel 0 and 1 are available for this command)

- **Draw Ellipse**

This command is used to draw an ellipse with X radius (Ra), Y radius (Rb) and centered at (X0,Y0).

CMD Code	0x10
CMD format	DVID,0x11,0x10,X0_H,X0_L,Y0_H,Y0_L,Ra,Rb,0x00,,0x00,0x00,0x00,0x00,0x00,Visible,0x00,FillStyle ,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

X0: 0 to LCDResX –1

Y0: 0 to LCDResY – 1

Ra,Rb: 0 to 255 (Ra and Rb must be less than LCDResX –1 and LCDResY – 1)

Visible: 0 = The Ellipse is not visible (clear the Ellipse)

1 = The Ellipse is visible (draw the Ellipse)

FillStyle: see fill style table (only FillStyel 0 and 1 are available for this command)

- **Draw Rectangle**






This command is used to draw a rectangle with upper left corner (x1,y1) and lower right corner (x2,y2).

CMD Code	0x0E
CMD format	DVID,0x11,0x0E,X1_H,X1_L,Y1_H,Y1_L,X2_H,X2_L,Y2_H,Y2_L,0x00,0x00,0x00,







	0x00,0x00,Visible,DrawStyle,FillStyle,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

X1: 0 to LCDResX – 1
 Y1: 0 to LCDResY – 1
 X2: 0 to LCDResX – 1
 Y2: 0 to LCDResY – 1
 X2 > X1
 Y2 > Y1
 Visible: 0 = The Rectangle is not visible (clear the rectangle)
 1 = The Rectangle is visible (draw the rectangle)
 DrawStyle: see Draw style table
 FillStyle : see Fill style table

Draw Style Table

Draw Style name	Code	
Normal	0x00	
Bold	0x01	
Thick	0x02	
Dot	0x03	
Dash	0x04	

Fill Style Table

Fill Style name	Code	
Transparent (clear inside)	0x00	
Solid	0x01	
Horizontal lines	0x02	
Vertical lines	0x03	
Cross	0x04	
No Fill	0x05	

• **Write Text**

This command is used to write one line of text starting at (X,Y) and with the selected Font.

CMD Code	0x11
CMD format	DVID,LB,0x11,X_H,X_L,Y_H,Y_L,Font,Asc1,Asc2.....AscN,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

X: 0 to LCDResX – 1
 Y: 0 to LCDResY – 1

Font: see Font table below

Font name	Cell Size	Font Code	Number of Characters
Font1	8X8	1	128
Font2	8X8	2	128
Font3	8X16	3	128
Font4	8X16	4	128
Font5	16X24	5	128
Font6	16X24	6	128

Asc1..AscN: Characters Ascii codes

N: is the number of characters (N <= 40)

LB = 6 + N

Example: To write "Hello" to the screen at (0,0) with Font 1

CMD Format = DVID,0x09,0x11,0x00,0x00,0x00,0x00,0x01,0x48,0x65,0x6C,0x6C,0x6F,CS

X Y Font1 H e l l o

- **Show Bitmap**

This command is used to show (display) a bitmap or image already saved in the module EEPROM starting at (X,Y).

CMD Code	0x12
CMD format	DVID,0x07,0x12,X_H,X_L,Y_H,Y_L,BMCode,BMID,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

X: 0 to LCDResX - 1

Y: 0 to LCDResY - 1

BMCode: Bitmap Code (see bitmap table)

BMID: Bitmap ID number

Bitmap Table

Bitmap Size/Name	Code	ID
Image	0x01	0 - 2
64X64	0x02	0 - 31
48X48	0x03	0 - 31
32X32	0x04	0 - 31
24X24	0x05	0 - 31
16X16	0x06	0 - 31
8X8	0x07	0 - 31
8X16	0x08	0 - 31
16X8	0x09	0 - 31
32X16	0x0A	0 - 31
16X32	0x0B	0 - 31
Extra 32X32	0x0C	0 - 63
Digit Small	0x14	0 - 23
Digit Medium	0x15	0 - 23
Digit Big	0x16	0 - 23
Digit Huge	0x17	0 - 23

- **Config TS**

This command is used to configure the touch screen by writing to the touch screen setting byte.

CMD Code	0x19
CMD format	DVID,0x02,0x19,TSConfigByte,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

- **CAL TS**

This command is used to calibrate the touch screen. Two points touch screen calibration algorithm is used.

CMD Code	0x1B
CMD format	DVID,0x01,0x1B,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

- **Get Touch Data**

This command is used to read the touch coordinates and status.

CMD Code	0x1A
CMD format	DVID,0x01,0x1A,CS
ACK format	DVID,0x06,0x09, TX_H, TX_L, TY_H, TY_L, TZ1_H, TZ1_L, TZ2_H, TZ2_L, TStatus, CS
NAK format	DVID,0x15,CS

TX_H&TX_L: Touch X coordinate

TY_H&TY_L: Touch Y coordinate

TZ1_H&TZ1_L: Touch Z1 value

TZ2_H&TZ2_L: Touch Z2 value

H: MSB

L: LSB

- **Get CAL Constants**

This command is used to read the calibration constants that being calculated after a successful touch screen calibration. The CAL constants are saved into the controller EEPROM and they will restore on power-on.

CMD Code	0x1D
CMD format	DVID,0x01,0x1D,CS
ACK format	DVID,0x06,0x08, A_H, A_L, B_H, B_L, C_H, C_L, D_H, D_L, CS
NAK format	DVID,0x15,CS

A_H&A_L: Cal constant A

B_H&B_L: Cal constant B

C_H&C_L: Cal constant C

D_H&D_L: Cal constant D

2- Objects overview:

Objects are the building blocks for any graphical user interface application. A list of 20 precoded objects is included to ease the GUI application development. Each object has a unique Code and ID number, both numbers are assigned automatically by the GooyLCD IDE. The number of objects that can be added to a screen is limited by ObjRAM (Object RAM) and the LCD view area. The following table shows the object codes and the maximum number per screen.

Object Name	Code	Max/Screen	ID Range
Screen	-	-	0 - 23
Form	0x01	5	0 - 4
Label	0x02	10	0 - 9
TextBox	0x03	8	0 - 7
CheckBox	0x04	8	0 - 7
Button	0x05	16	0 - 15
RadioButton	0x06	8	0 - 7
TextField	0x07	16	0 - 15
ListBox	0x08	1	0
Bitmap	0x0A	16	0 - 15
Line	0x0B	16	0 - 15
Circle	0x0C	8	0 - 7
Ellipse	0x0D	8	0 - 7
Rectangle	0x0E	16	0 - 15
ImageSeq	0x0F	8	0 - 7
Animation	0x10	8	0 - 7
BarGraph	0x14	8	0 - 7
Chart	0x15	3	0 - 2
NumberBox	0x16	8	0 - 7
LinearGage	0x17	8	0 - 7
Slider	0x18	8	0 - 7

The object properties are divided into two groups:

- Design Time Properties: Each object has its own design (development) time properties. Design Time properties are fully customized through the Object property window (see “GooyLCD IDE user manual for more information). The following table describes all design time properties:

Property name	Description
Code	Object code
ID	Object ID
X	Object position X coordinate
Y	Object position Y coordinate
Width	Object width in pixels
Height	Object height in pixels
BStyle	Object boarder style
Font	Object font
MaxLenght	Object maximum number of characters
TextOffset	Object text offset
NOItem	Object number of items
BMCode	Bitmap code
BMID	Bitmap ID
P1(X1,Y1)	Object drawing Point 1
P2(X2,Y2)	Object drawing Point 2
Ra	Object radius Ra
Rb	Object radius Rb

NOImage	Object number of images
ConfigByte	Object configuration byte
NOTick	Object number of ticks
TickFreq	Object ticks frequency
Visible	Object is visible/unvisible (available at Run time)
DrawStyle	Object Draw style (available at Run time)
FillStyle	Object Fill style (available at Run time)
Text	Object text (available at Run time)
Cursor	Cursor is enabled or disabled (available at Run time)

- Run Time Properties: Each object also has its own Run Time properties. The object commands are used by the host controller to update the object run time properties.

Examples:

Let us assume that Screen0 is already loaded and you need to change the caption of a Label. The label code is “2” and its ID is “5”.

To change the label caption to “Error”, your host controller needs to send the object command “Text” as following:

Send_ObjCmd_Text(0x02, 0x05, “E”, “r”, “r”, “o”, “r”)

To update a BarGraph with ID = 1 to new value (0x40):

Send_ObjCmd_Value(0x14, 0x01, 0x40)

To read the BarGraph status byte

Send_ObjCmd_Status(0x14,0x01)

To Redraw an Animation (ID = 1)

Send_ObjCmd_Trigger(0x10,0x01)

To change a button (ID = 1) state to Unpressed

Send_ObjCmd_Click(0x05,0x01,0x00)

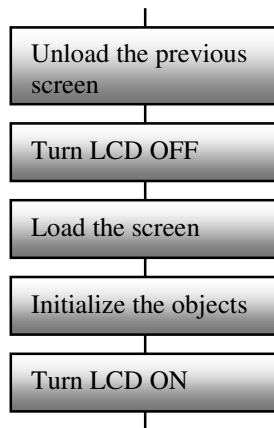
The following table describes all runtime properties and the objects how use them.

Property Name	Object	Description
HighLight	Label, TextBox, CheckBox, RadioButton, Button, Bitmap	To highlight or unhighlight text or graphics (Bitmap)
Visible	All objects except TextField, Animation, ListBox	To show or hide an object
Check	CheckBox, RadioButton	To switch between checked and unchecked states
Click	Button	To switch between clicked and unclicked states
Cursor	TextBox	To enable or disable the cursor
DrawStyle	Line, Rectangle	To change the draw style
FillStyle	Circle, Ellipse, Rectangle	To change the fill style
Pixel	Chart	To update the chart with new pixel(x,y)
Value	BarGraph, ImageSeq, LinearGage Slider	To update an object with a new value
Scroll	ListBox	To scroll up or down
Trigger	Animation	To update an animation
Clear	Label, TextBox, Chart	To clear text or graphics
Status	All Objects	To get an object status
Touch	All Objects except TextField, Animation	To check if an object has been touched
Text	Label, TextBox, NumberBox	To write string or number to an object

3- Objects descriptions:

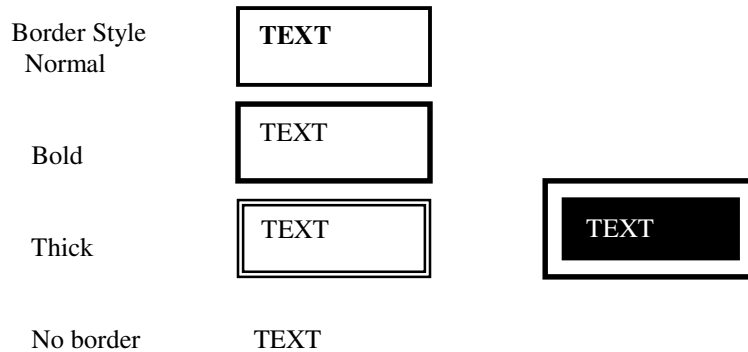
17.1 Screen:

Screen is not an object, it is a container for the objects shown on the display (like a form in visual programming). The screen has the same resolution as the target display and only one screen can be displayed at any time. A GUI program usually contains more than one screen, the first screen shown after power on is usually the main screen. From this screen, the user will navigate through the entire GUI program. Up to 24 screens can be added to GUI program which, is more than enough for most application. Two object commands are used to load and show or unload and clear a screen. When a screen is loaded, the object parameters are transferred from the controller GUI EEPROM to the ObjRAM and at the same time drawn on the display screen. When a screen is unloaded, the display screen and ObjRAM will be cleared. So it necessary when you load a new screen, to unload the previous one first. Immediately after loading a screen, the host controller needs to initialize the objects (similar to Form_Load method) which can take some time depending on the number of objects and the initialization procedure. If you have complex screens and you like to hide the screens initialization, you can use the terminal command (SetLCDMode) to turn the LCD OFF and then turn it ON after the initialization is done



17.2 Label:

Label is used to place text anywhere within the view area of the display. The text is editable at run time. Also, you can set the Label object to act as TextField object (Fixed Text) while you still able to use the other label run time properties



Runtime Properties:

- Highlight
- Text (Has no effect if the Label is set to act as TextField)
- Visible
- Touch
- Status

Note: You need to write the Label text every time you need to highlight/unhighlight or hide/show the label.

17.3 TextBox:

TextBox is used to place text anywhere within the view area of the display. It looks and behaves like label except you can enable or disable a vertical text cursor. The text is editable and the cursor can be enabled or disabled at run time. Also, you can set the TextBox to act as TextField (fixed text) while you are still able to use the other TextBox run time properties.



The cursor is 1 pixel wide vertical line, the height of the cursor is automatically adjusted according to the selected font height. Every time a character is added or deleted from a textbox, the cursor moved accordingly (left or right). The cursor does not blink. However, the host controller can use cursor enable/disable to blink the cursor at certain rate and implement “Get Focus, Loss Focus” functions when more than one textbox are shown on the screen.

Runtime Properties:

- Highlight
- Text (Has no effect if the TextBox is set to act as TextFiled)
- Visible
- Cursor (Has no effect if the TextBox is set to act as TextFiled)
- Touch
- Status

Note: You need to write the text every time you need to highlight/unhighlight or hide/show a TextBox.

17.4 TextField:

TextField is used to place static text anywhere within the view area of the display screen. The text is not editable at run time. TextField has no runtime properties.

17.5 Button:

Button is a control object used to perform a function or set of functions when is clicked. The Button style can be Standard (windows style with the button caption displayed at the center) or Graphical (one bitmap for Unpressed state and other for Pressed state).

Standard Button



Unclicked State

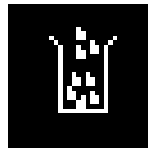


Clicked State

Graphical Button



Unpressed State



Pressed State

Runtime Properties:

- HighLight
- Click
- Visible
- Touch
- Status

17.6 CheckBox:

CheckBox is a special type of button that indicates if a state is true or false (1 or 0). It has two states: Checked (Selected) or Unchecked(Unselected).

Average Data

Average Data

Runtime Properties:

- HighLight
- Check
- Visible
- Touch
- Status

17.7 RadioButton:

RadioButton is also a special type of button that allows the selection of one item in list. The host controller need to group the radio buttons into a list and insure that each selection clears the previous choice.

CH0

CH1

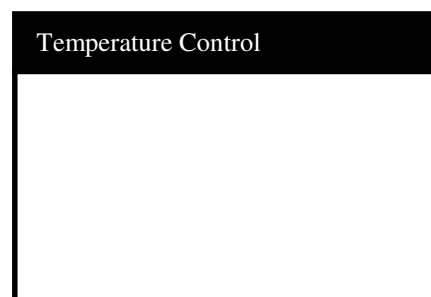
CH2

Runtime Properties:

- HighLight
- Check
- Visible
- Touch
- Status

17.8 Frame:

Frame is used to display a group of objects under one title.

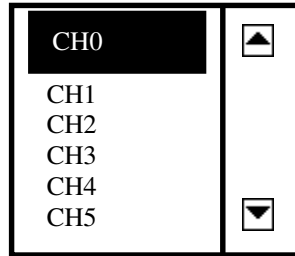


Runtime Properties:

- Visible
- Touch
- Status

17.9 ListBox:

ListBox allows the selection of one item from a list. All items in the list are visible and the selected item is highlighted by reversing the drawing color of the item. The ScrollUp and ScrollDown bitmaps are OEM bitmaps, which are 8X8 and saved at their predefined locations in the controller EEPROM.

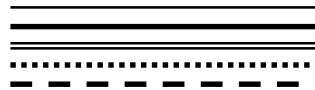


Runtime Properties:

- Scroll
- Status

17.10 Line:

This object is used to place a line anywhere within the view area of the display. Line draw style is editable at run time.

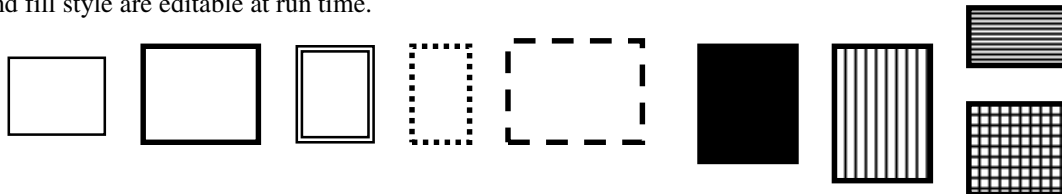


Runtime Properties:

- DrawStyle
- Visible
- Touch
- Status

17.11 Rectangle:

This object is used to place a rectangle anywhere within the view area of the display. Rectangle draw style and fill style are editable at run time.



Runtime Properties:

- DrawStyle
- FillStyle
- Visible
- Touch
- Status

17.12 Circle:

This object is used to place a circle anywhere within the view area of the display. Circle fill style is editable at run time.

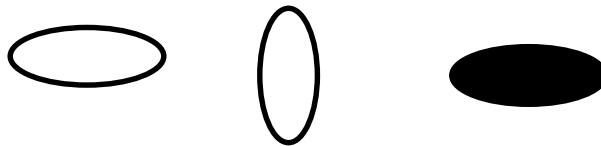


Runtime Properties:

- FillStyle
- Visible
- Touch
- Status

17.13 Ellipse:

This object is used to place an ellipse anywhere within the view area of the display. Ellipse fill style is editable at run time.

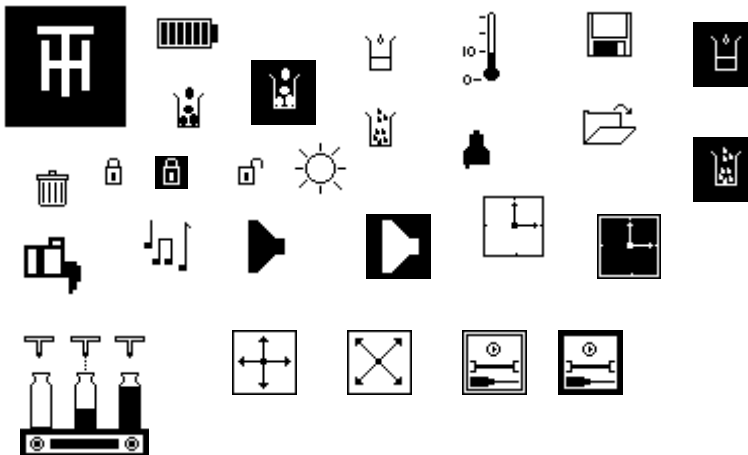


Runtime Properties:

- FillStyle
- Visible
- Touch
- Status

17.14 Bitmap:

This object is used to place a bitmap anywhere within the view area of display. It allows you to create a graphical menu or list, custom button or, flashing warning or alarm.










Runtime Properties:

- HighLight
- Visible
- Touch
- Status



17.15 ImageSequence:

This object is used to display a bitmap from an array of bitmaps stored in sequence in the font/bitmap EEPROM. It allows you to graphically show or monitor a process like charging battery or to implement custom button with two states or more.

Example 1: charging battery

Value	Bitmap
0	
1	
2	
3	
4	
5	
6	

Example 2: custom button

Value	Bitmap
0	
1	

Runtime Properties:

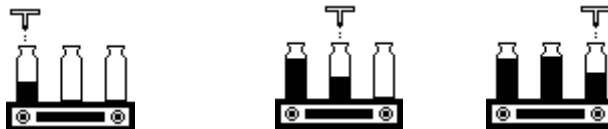
- Visible
- Value
- Touch
- Status

17.16 Animation:

Animation is a process in which you rapidly present a series of bitmaps (frames), to create the illusion of motion. If you present the bitmaps quickly enough, and if the changes from one bitmaps to the next are not too large, an observer perceives the series as single bitmaps that is changing over time.

This object can operate in 2 different modes:

- Static: the animating frames (bitmaps) do not move (change position). This mode uses clear and draw method to display an array of bitmaps one after another in a loop. Every time the animation object is triggered by the host controller, the previous frame is cleared and the next frame is drawn at the same position creating an animating motion of bitmaps with speed controlled by the triggering rate.



- Moving: the animating frames (bitmaps) move horizontally or vertically in 4 user defined direction: Left To Right, Right To Left, Up To Down, and Down To Up. This mode is similar to the static mode except, the next frame is drawn at a position defined by the direction of motion and the dimension of the bitmap.

Trigger	Left To Right [Persist = 1]
0	➡
1	➡ ➡
2	➡ ➡ ➡
3	➡ ➡ ➡ ➡
4	➡

Trigger	Left To Right [Persist = 0]
0	➡
1	➡
2	➡
3	➡
4	➡

Runtime Properties:

- Trigger
- Status

17.17 Chart:

Chart is used to represent data as a function of independent X variables, where each Y value provides a data point for each X value. Three charts can be added to a screen and each chart can only display one series. Chart can display data in two formats:

- XY format: The chart is updated using the X coordinate (X value) and Y coordinate (Y value) of the pixel.
- YT format: The chart is updated using the Y coordinate (Y value) of the pixel while the X value (time axis) is automatically incremented every time a new pixel is received.

The Chart can be drawn as dots (Dot Style) or as lines (Vector Style).

Also, a new object command (Array) is added to fill the chart with an array of points. See the object command section for more information on how to use this command.

Chart object can operate in two different modes:

- Digital: In this mode, the "Pixel" object command is used to update the chart. If the chart format is XY, then both X value and Y value are used to draw the pixel while, if the chart format is YT, then only the Y value is used to draw the pixel.
- Analog or "Oscilloscope Mode": Three analog inputs (AIN0, AIN1, and AIN3) and Horizontal Scanning Signal (HSS) are used to update the chart in either format. At the rising edge of HSS, the input analog signal is converted to digital using the internal 10 bits A/D converter, 4 samples are averaged and the result is scaled by 8 (ADC(AINx)) to give the Y axes a resolution of 7 bits (128 level) with dynamic input range from 0 to 5V.

If the chart format is YT, then the chart Y axes can be assigned to AIN0, AIN1 or AIN2 and Y value is calculated using the formula:

$$Y \text{ value} = \text{ADC}(\text{AIN}_{0,1,2}) * Y_{\text{max}}/128$$

X value is incremented by 1.

If the chart format is XY, then the chart Y axis can be assigned to AN1 or AN2, X axes is fixed to AIN0. Y value and X value are calculated using the following formulas:

$$Y \text{ value} = \text{ADC}(\text{AIN}_{1,2}) * Y_{\text{max}}/128$$

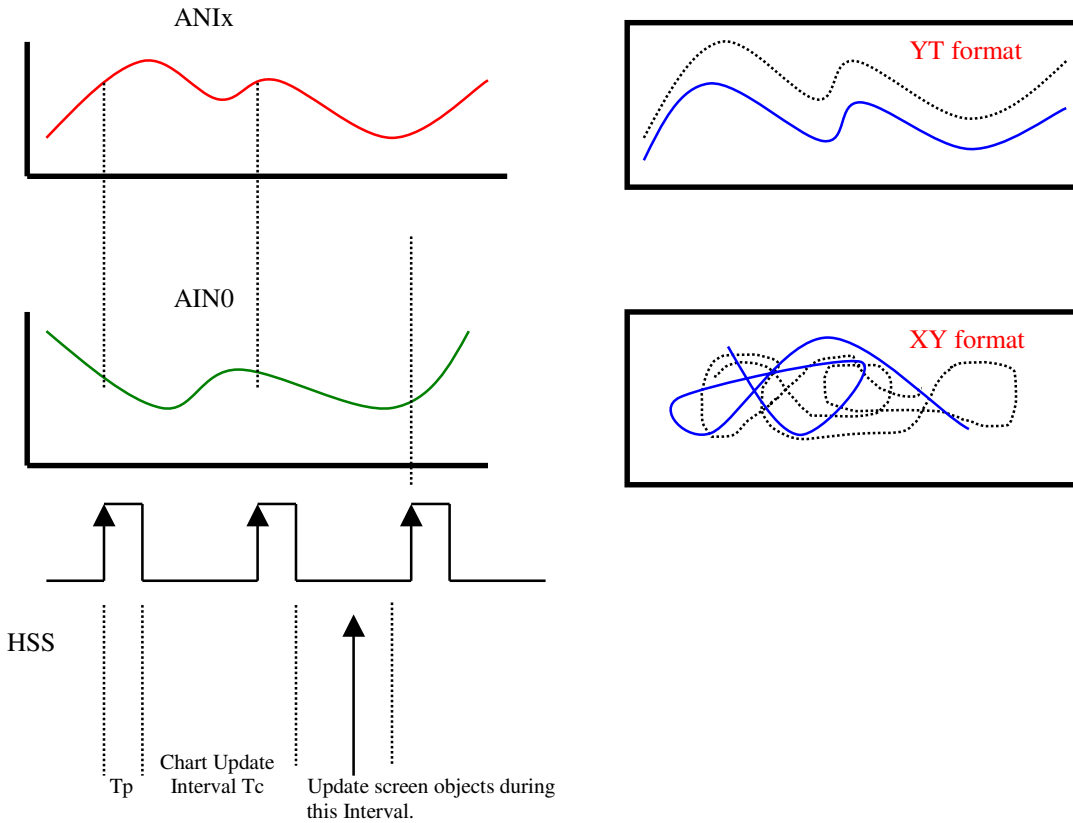
$$X \text{ value} = \text{ADC}(\text{AIN}_0) * X_{\text{max}}/128$$

Where:

$$Y_{\text{max}} = \text{Chart Height} - 1$$

$$X_{\text{max}} = \text{Chart Width} - 1$$

ADC(AIN_x): is the result (after scaling) of the A/D converter for particular analog input (AIN_x)



The chart update rate in the analog mode is faster than in the digital mode which is necessary for some applications like medical devices.

In analog mode, the chart update rate “fc” is the frequency of HSS and is given by:

$$f_c = 1/T_c \text{ in HZ}$$

The maximum fc value depends on Ymax and the number of charts. For best results, adjust HSS frequency to give you the desired performance.

HSS pulse width “Tp” should be from 1 to 2msec and it is recommended to update the screen objects immediately after the falling edge of HSS to avoid communication errors with the module.

Runtime Properties:

- Array
- Pixel
- Clear
- Touch
- Status

The analog input voltage must be in the range from 0V to 5.0V. A value beyond this range can cause a permanent damage to the module and/or the display.

17.18 BarGraph:

BarGraph is a valuable indicator for process control application. The value is represented by the fill level within an outer body. The update is fixed to Smooth and from left to right for horizontal orientation or from down to up for vertical orientation.



Runtime Properties:

- Value
- Visible
- Touch
- Status

17.19 NumberBox:

NumberBox is a great object to display numbers. It does mimic a standard seven-segment display and able to display up to 8 digits at four different digit sizes. The digit bitmaps are user editable.

The provided digit sizes are:

- Small: 8X16 cell size
- Medium: 16X20 cell size
- Big: 24X40 cell size
- [Huge: 64X128 cell size](#)

12:34:45

100.34845

+125.901

The following table shows the digit bitmaps (small, medium, big or huge) and their ID numbers. They must be saved into the controller EEPROM at their predefined locations.

Digit bitmap	ID number	Ascii code
0	0	0x30
1	1	0x31
2	2	0x32
3	3	0x33
4	4	0x34
5	5	0x35
6	6	0x36
7	7	0x37
8	8	0x38
9	9	0x39
0.	10	
1.	11	
2.	12	
3.	13	
4.	14	
5.	15	
6.	16	
7.	17	
8.	18	
9.	19	
+	20	0x2b
-	21	0x2d
blank	22	0x20
:	23	0x3a

NumberBox uses the “Text” object command to write the number.

Example 1: To write this number (+12.78) to a numberbox

+ 1 2 . 7 8
 0x2b,0x31,0x32,0x2e,0x37,0x38

Example 2: To clear a 4 digits numberbox

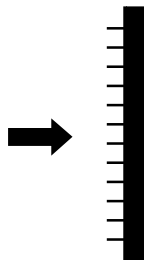
0x20,0x20,0x20,0x20

Runtime Properties:

- Text
- Status

17.20 LinearGage:

LinearGage is another valuable indicator for process control application, where the value is represented by moving pointer along the gage body. The pointer bitmap is user editable.

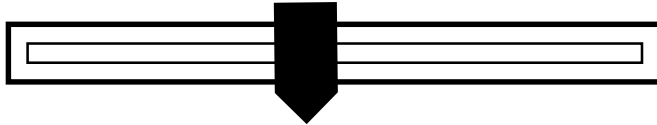


Runtime Properties:

- Visible
- Value
- Touch
- Status

17.21 Slider:

Sliders provide a visual method for numeric input. The handle bitmap is user editable. This object is typically used with the touch screen to change the value of parameter or to control a process.



Runtime Properties:

- Visible
- Value
- Touch
- Status

18. Object Commands:

- **HighLight:**

This command is used to highlight or unhighlight text or graphics.

CMD Code	0x46
CMD format	DVID,0x04,0x46,ObjCode,ObjID,CMDByte,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ObjCode: Object Code

ObjID: Object ID number

CMDByte: Command Byte

CMDByte = 0 => Unhighlight

CMDByte = 1 => Highlight

- **Check**

This command is used to check or uncheck checkbox or radiobutton.

CMD Code	0x47
CMD format	DVID,0x04,0x47,ObjCode,ObjID,CMDByte,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ObjCode: Object Code

ObjID: Object ID number

CMDByte: Command Byte

CMDByte = 0 => Uncheck

CMDByte = 1 => Check

- **Click**

This command is used to change the state of button from click to unclick or vice versa.

CMD Code	0x4B
CMD format	DVID,0x04,0x48,ObjCode,ObjID,CMDByte,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ObjCode: Object Code

ObjID: Object ID number

CMDByte: Command Byte

CMDByte = 0 => Unclick

CMDByte = 1 => Click

- **Cursor**

This command is used to enable or disable the cursor.

CMD Code	0x48
CMD format	DVID,0x04,0x48,ObjCode,ObjID,CMDByte,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ObjCode: Object Code

ObjID: Object ID number

CMDByte: Command Byte

CMDByte = 0 => Disable
CMDByte = 1 => Enable

- **Visible:**

This command is used to show or hide an object.

CMD Code	0x4E
CMD format	DVID,0x04,0x4E,ObjCode,ObjID,CMDByte,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ObjCode: Object Code
ObjID: Object ID number
CMDByte: Command Byte

CMDByte = 0 => Hide
CMDByte = 1 => Show

- **Scroll:**

This command is used to scroll up or down.

CMD Code	0x50
CMD format	DVID,0x04,0x50,ObjCode,ObjID,CMDByte,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ObjCode: Object Code
ObjID: Object ID number
CMDByte: Command Byte

CMDByte = 0 => Scroll Up
CMDByte = 1 => Scroll Down

- **Value:**

This command is used to update an object parameter value.

CMD Code	0x51
CMD format	DVID,0x04,0x51,ObjCode,ObjID,Value,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ObjCode: Object Code
ObjID: Object ID number

Value: single byte from 0 to 255

- **Text:**

This command is used to write a text to an object.

CMD Code	0x49
CMD format	DVID,LB,0x49,ObjCode,ObjID,CHR1,CHR2CHRn,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ObjCode: Object Code
ObjID: Object ID number
CHR1CHRn: Ascii codes

LB = 3 + n

- **DrawStyle:**

This command is used to change the draw style of an object.

CMD Code	0x53
CMD format	DVID,0x04,0x53,ObjCode,ObjID,DrawStyleCode,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ObjCode: Object Code

ObjID: Object ID number

DrawStyleCode: Draw Style Code (See drawstyle codes table)

- **FillStyle:**

This command is used to change the fill style of an object.

CMD Code	0x54
CMD format	DVID,0x04,0x54,ObjCode,ObjID,FillStyleCode,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ObjCode: Object Code

ObjID: Object ID number

FillStyleCode: Fill Style Code (See fill style codes table)

- **Pixel:**

This command is used to update the chart with new (X,Y) point .

CMD Code	0x55
CMD format	DVID,0x05,0x55,ObjCode,ObjID,X ,Y,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ObjCode: Object Code

ObjID: Object ID number

X: X value

Y: Y value

- **Array:**

This command is used to fill the chart with an array of points. The array holds 64 (X,Y) points which can be used to fill the chart in Digital YT or Digital XY modes only. If the chart Xmax is higher than 64, then multiple Array commands are needed to fill the chart.

CMD Code	0x58
CMD format	DVID,0x85,0x58,ObjCode,ObjID, ClearChartFirst, X0 ,Y0, X1, Y1,,X63, Y63 CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ObjCode: Chart Object Code

ObjID: Chart Object ID number

ClearChartFirst: This byte determine if you need to clear the chart and start from the chart origin or not. The first Array command must have ClearChartFirst set to "0" to clear the chart and start the filling from the chart origin. If more array commands are needed to fill the chart, the subsequent commands must have their ClearChartFirst bytes set to "1" to start the filling from the current (X,Y).

ClearChartFirst = 1 => Do not clear the chart and start filling from the current (X,Y)

ClearChartFirst = 0 => Clear the chart first and start filling from the origin (0,0)

X: X value (This value is not important in YT mode)

Y: Y value

For example: If you have a chart with Xmax = 96 (95 points are valid), then 2 array commands are needed to fill the chart.

The first command will hold the first 64 points with ClearChartFirst is cleared while, the second command will hold the remaining 31 points (the other 33 points are not used) with ClearChartFirst is set.

- **Clear:**

This command is used to clear text or graphics.

CMD Code	0x4A
CMD format	DVID,0x03,0x4A,ObjCode,ObjID,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ObjCode: Object Code

ObjID: Object ID number

- **Trigger:**

This command is used to update the animation object.

CMD Code	0x52
CMD format	DVID,0x03,0x52,ObjCode,ObjID,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ObjCode: Object Code

ObjID: Object ID number

- **Status:**

This command is used to read the object status.

CMD Code	0x4F
CMD format	DVID,0x03,0x4F,ObjCode,ObjID,CS
ACK format	DVID,0x06,0x01,ObjStatus,CS
NAK format	DVID,0x15,CS

ObjCode: Object Code

ObjID: Object ID number

ObjStatus

7	6	5	4	3	2	1	0
Err	x	C	LL	UL	H	S	V

Bit0: Visible

0 = Object is not visible

1 = Object is visible

Bit1: State

0 = Object state is unclicked or unchecked

1 = Object state is clicked or checked

Bit2: Highlight

0 = Object is not highlighted

1 = Object is highlighted

Bit3: Upper Limit

0 = Object has not reached its upper limit

1 = Object has reached its upper limit

Bit4: Lower Limit

0 = Object has not reached its lower limit
 1 = Object has reached its lower limit

Bit5: Cursor
 0 = Cursor is OFF
 1 = Cursor is ON

Bit6: is not used

Bit7: Error
 0 = Object error has occurred
 1 = No object error

• **Touch:**

This command is used to check if an object is being touched by the user or to scan a group of objects with the same code to check if any of them is being touched. This command returns touch status (Tstatus), touch coordinates (X,Y), touched object code and ID. If no object is being touched, touch coordinates, code and ID will read 0. To check a single object, set ObjID-start = ObjID-End = Object ID.

CMD Code	0x57
CMD format	DVID,0x04,0x57,ObjCode,ObjID-Start,ObjID-End,CS
ACK format	DVID,0x06,0x07,Tstatus,TObjCode,TObjID, TX_H, TX_L, TY_H, TY_L, CS
NAK format	DVID,0x15,CS

ObjCode: Object Code
 ObjID-Start: Objects group starting ID number
 ObjID-End: Objects group ending ID number
 TStatus: Touch Status (See Touch Screen section for more information)
 TobjCode: Touched Object Code
 TobjID: Touched Object ID number
 TX_H&TX_L: Touch X coordinate
 YT_H&TY_L: Touch Y coordinate

• **Load Screen:**

This command is used to load and display a screen.

CMD Code	0x3E
CMD format	DVID,0x02,0x3E,ScreenID,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ScreenID: Screen ID number (0 to 23)

• **Unload Screen:**

This command is used to unload and clear a screen.

CMD Code	0x3F
CMD format	DVID,0x02,0x3F,ScreenID,CS
ACK format	DVID,0x06,0x00,CS
NAK format	DVID,0x15,CS

ScreenID: Screen ID number (0 to 23)

Terminal Commands Table

Command	Code in HEX	Code in DEC
Reset	0x01	1
Set Configuration	0x05	5
Get Module Info	0x02	2
Set LCD Mode	0x06	6
Adjust Contrast	0x15	21
Adjust Brightness	0x16	22
Backlight ON/OFF	0x1C	28
Set Alarm	0x17	23
Beep	0x18	24
Clear Screen	0x0B	11
Draw Pixel	0x0C	12
Draw Line	0x0D	13
Draw Circle	0x0F	15
Draw Ellipse	0x10	16
Draw Rectangle	0x0E	14
Write Text	0x11	17
Show Bitmap	0x12	18
Config TS	0x19	25
CAL TS	0x1B	27
Get Touch Data	0x1A	26
Get CAL Constants	0x1D	29

Object Commands Table

Command	Code in HEX	Code in DEC
Highlight	0x46	70
Check	0x47	71
Click	0x4B	75
Cursor	0x48	72
Visible	0x4E	78
Scroll	0x50	80
Value	0x51	81
Text	0x49	73
DrawStyle	0x53	83
FillStyle	0x54	84
Pixel	0x55	85
Array	0x58	88
Clear	0x4A	74
Trigger	0x52	82
Status	0x4F	79
Touch	0x57	87
Load Screen	0x3E	62
Unload Screen	0x3F	63

Hardware Limited Warranty

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

Returns and Repair Policy

No merchandise may be returned for credit, exchange, or service without prior authorization from. To obtain warranty service, contact the factory and request an RMA (Return Merchandise Authorization) number. Enclose a note specifying the nature of the problem, name and phone number of contact person, RMA number, and return address. Authorized returns must be shipped freight prepaid to Haidar Technology LLC. 5837 Karric Square Drive, Dublin, OH 43016 with the RMA number clearly marked on the outside of all cartons. Shipments arriving freight collect or without an RMA number shall be subject to refusal. Haidar reserves the right in its sole and absolute discretion to charge a 15% restocking fee, plus shipping costs, on any products returned with an RMA.

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