4D LABS SEMICONDUCTORS

Workshop4

Integrated Development Environment

ViSi-Genie Reference Manual

Document Revision: 2.3

Document Date: 30th May 2019

Table of Contents

1. Introduction to ViSi-Genie

The **ViSi-Genie** is a breakthrough in the way 4D Labs' processors are programmed, it provides an easy method for designing complex Graphics User Interface applications without any coding. It is an environment like no other, a code-less programming environment that provides the user with a rapid visual experience, enabling a simple GUI application to be 'designed' from scratch in literally seconds.

ViSi-Genie does all the background coding, no 4DGL to learn, it does it all for you.

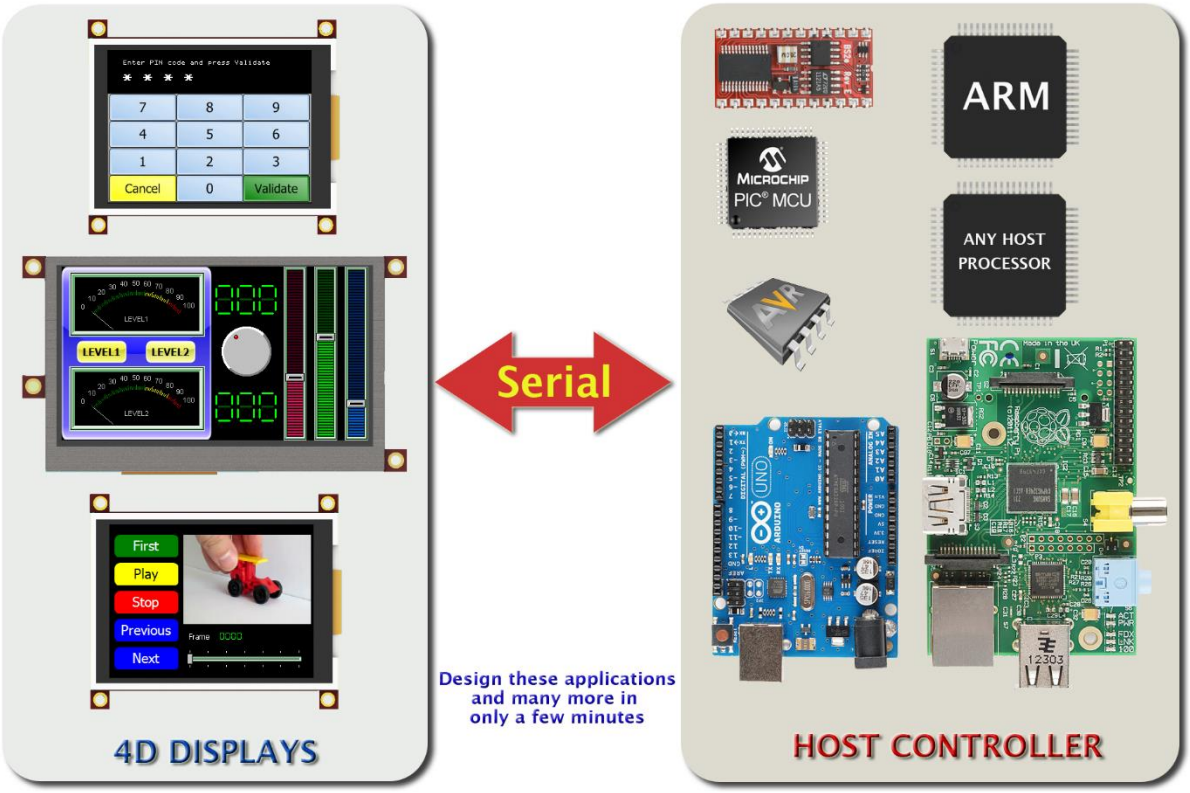
Pick and choose the relevant objects to place on the display, much like the ViSi environment, yet without having to write a single line of code. The full animation of the objects is done under-the-hood, such as pressing a button or moving the thumb of the slider. Each object has parameters which can be set, and configurable events to animate and drive other objects or communicate with an external host.

Simply place an object on the screen, position and size it to suit, set the parameters such as colour, range, text, and finally select the event you wish the object to be associated with, it is that simple.

In seconds you can transform a blank display into a fully animated GUI with moving meters, animated press and release buttons, and much more. **All without writing a single line of code!**

ViSi-Genie provides the user with a feature rich rapid development environment, second to none.

This document covers the ViSi-Genie functions available for the Picaso, Picaso-lite and Diablo16 Processors. This document should be used in conjunction with the *ViSi-Genie User Guide*.



2. Graphics Build and Display Order

When a form is built the background image (of just the colour if it doesn't have an image) and any 'backgrounds' are 'built' first. Then each other object is 'turned on', one at a time and built, this is so that transparent parts of these objects will have the background image appearing in their transparent parts. After each object is built it is turned off again.

When a form is activated the background is drawn, then any 'primitives' and finally the other graphical objects. The primitives and graphic objects are drawn in the order they are added to the form.

In general it is not a good idea to overlap objects, however, this can be used to create various effects as overlapping objects are 'brought to the front' whenever they are written to by the host.

3. Communication Protocol

The ViSi-Genie display platform offers a serial communications protocol called the **Genie Standard Protocol**.
The protocol provides access to a majority of the display's features and gives the host detailed information on the current state of all the objects used in the display application.

The **Genie Standard Protocol** provides a simple yet effective interface between the display and the host controller and all communications are reported over this bidirectional link. The protocol utilises only a handful of commands and is simple and easy to implement.

Serial data settings are:

8 Bits, No Parity, 1 Stop Bit.

The baud rate for the display is selected from the Workshop4 Genie project. The user should match the same baud rate on the host side.

Note: RS-232 handshaking signals (i.e., RTS, CTS, DTR, and DSR) are not supported by the ViSi-Genie protocols. Instead, only the RxD (received data), TxD (transmitted data), and signal ground are used.

3.1. Genie Standard Protocol

This section describes the Genie Standard Protocol in detail.

3.1.1. Protocol Definitions

The commands and parameters are sent and received using a very simple messaging structure. The message consists of a command byte, command parameters, and a checksum byte. The checksum ensures some the integrity of the message. The following figure shows the organisation of the message.

CMD	PARAM (1 to N bytes)	CHKSUM

- **CMD:** This byte indicates the command code. Some commands will have more parameters than others. The table below outlines the available commands and their relevant parameters.
- **PARAM:** Parameter bytes (variable); a variable number of parameter bytes (between 1 to N) that contains information pertaining to the command. Refer to the command table below.
- **CHKSUM:** Checksum byte; this byte is calculated by taking each byte and XOR'ing all bytes in the message from (and including) the CMD byte to the last parameter byte. Then, the result is appended to the end to yield the checksum byte.

Note: If the message is correct, XOR'ing all the bytes (including the checksum byte) will give a result of zero.

3.1.2. Protocol Definitions

Command	Code	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter N	Checksum
READ_OBJ	**0x00**	Object ID	Object Index	-	-	-	Checksum
WRITE_OBJ	**0x01**	Object ID	Object Index	Value (msb)	Value(lsb)	-	Checksum
WRITE_STR	**0x02**	String Index	String Length	String (1 byte chars)			Checksum
WRITE_ STRU	**0x03**	String Index	String Length	String (2 byte chars)			Checksum
WRITE_ CONTRAST	**0x04**	Value	-	-	-	-	Checksum
REPORT_OBJ	**0x05**	Object ID	Object Index	Value (msb)	Value(lsb)	-	Checksum
REPORT_EVENT	**0x07**	Object ID	Object Index	Value (msb)	Value(lsb)	-	Checksum
WRITE_MAGIC_BYTES	**0x08**	Object Index	Length	Array (1 byte values)			Checksum
WRITE_MAGIC_DBYTES	**0x09**	Object Index	Length	Array (2 byte values)			Checksum
REPORT_MAGIC_EVENT_BYTES	**0x0A**	Object Index	Length	Array (1 byte values)			Checksum
REPORT_MAGIC_EVENT_DBYTES	**0x0B**	Object Index	Length	Array (2 byte values)			Checksum

3.1.3. Command Set Messages

This section provides detailed information intended for programmers of the Host Controller. It contains the message formats of the commands that comprise the ViSi-Genie protocol. New commands may be added in future to expand the protocol.

Acknowledgement Bytes Table

ACK	Acknowledge byte (**0x06**); this byte is issued by the Display to the Host when the Display has correctly received the last message frame from the Host. The transmission message for this is a single byte: **0x06**
NAK	Not Acknowledge byte (**0x15**); this byte is issued by the receiver (Display or Host) to the sender (Host or Display) when the receiver has not correctly received the last message frame from the sender. The transmission message for this is a single byte: **0x15**

3.1.3.1. Read Object Status Message

Message	**CMD, OBJ-ID, OBJ-INDEX, CHKSUM**	
	CMD	**0x00** : READ OBJECT Command Code.
	OBJ-ID	Object ID. Refer to Object ID table for the relevant codes
	OBJ-INDEX	This byte specifies the index or the item number of the Object
	CHKSUM	Checksum byte
Direction	From **Host** to **Display**	
Length	Message length is **4** bytes	
Response	From **Display** to **Host**: NAK or REPORT OBJECT	
	NAK	If the Display did not understand the message it will respond with the NAK byte. In this case, the Host should retransmit the message.
	REPORT OBJ	If the Display understood the message, it will respond back with the Report Object Status message.
Description	The host issues the **Read Object** message when it wants to determine the current value of a specific object instance. Upon receipt of this message the display will reply with either a NAK (in the case of an error) or the REPORT_OBJ message (0x05, Object-ID, Object Index, Value{msb}, Value{lsb}, checksum). For more details refer to the *Report Object Status* message in this section.	
Example		

3.1.3.2. Write Object Value Message

Message	CMD, OBJ-ID, OBJ-INDEX, VALUE(msb), VALUE(lsb), CHKSUM	
	CMD	**0x01** : WRITE OBJECT Command Code
	OBJ-ID	Object ID. Refer to Object ID table for the relevant codes
	OBJ-INDEX	This byte specifies the index or the item number of the Object
	VALUE(msb)	Most significant byte of the 2 byte VALUE
	VALUE(lsb)	Least significant byte of the 2 byte VALUE
	CHKSUM	Checksum byte
Direction	From **Host** to **Display**	
Length	Message length is **6** bytes	
Response	From **Display** to **Host**, ACK or NAK	
	ACK	If the Display understood the message, it will respond back to the host with an ACK after completing the requested action.
	NAK	If the Display did not understand the message it will respond with a NAK. In this case, the Host should retransmit the message.
Description	The host issues the **Write Object** command message when it wants to change the status of an individual object item. For example, Meter 3 value needs to be set to 50.	
Example		

3.1.3.3. Write String (ASCII) Message

Message	CMD, STR-INDEX, STRLEN, "STRING", CHKSUM	
	CMD	**0x02** : WRITE STRING (ASCII) Command Code
	STR-INDEX	This byte specifies the index or the item number of the ASCII String Object
	STRLEN	Length of the string characters, including the null terminator
	STRING	ASCII String characters
	CHKSUM	Checksum byte
Direction	From **Host** to **Display**	
Length	Message length is **4** bytes + the number of string characters (including the null terminator).	
Response	From **Display** to **Host**: ACK or NAK	
	ACK	If the Display understood the message, it will respond back to the host with the ACK byte after completing the requested action.
	NAK	If the Display did not understand the message it will respond with the NAK byte. In this case, the Host should retransmit the message.
Description	A place holder for ASCII string objects can be defined and created in the Genie project. In order to display a dynamic string, the host can send this **Write String (ASCII)** message along with the string object index and then the string to be displayed. The maximum string length is 80 characters. **Note1:** The ASCII characters are 1 byte each. **Note2:** The String should be null terminated. **Note3:** Refer to the application notes for detailed information on Strings and their usage.	
Example		

3.1.3.4. Write String (Unicode) Message

Message	CMD, STR-INDEX, STRLEN, "STRING", CHKSUM	
	CMD	**0x03** : WRITE STRING (Unicode) Command Code
	STR-INDEX	This byte specifies the index or the item number of the Unicode String Object
	STRLEN	Length of the string characters, including the null terminator
	STRING	Unicode String characters (2 bytes per character).
	CHKSUM	Checksum byte
Direction	From **Host** to **Display**	
Length	Message length is **4** bytes + the number of string characters (including the null terminator).	
Response	From **Display** to **Host**: ACK or NAK	
	ACK	If the Display understood the message, it will respond back to the host with the ACK byte after completing the requested action.
	NAK	If the Display did not understand the message it will respond with the NAK byte. In this case, the Host should retransmit the message.
Description	A place holder for Unicode string objects can be defined and created in the Genie project. In order to display a dynamic string, the host can send this **Write String (Unicode)** message along with the string object index and then the string to be displayed. The maximum string length is 80 characters. **Note1:** The Unicode characters are 2 bytes each. **Note2:** The String should be null terminated. **Note3:** Refer to the 4D System's application notes for detailed information on Strings and their usage.	
Example		

3.1.3.5. Write Contrast Message

Message	CMD, VALUE, CHKSUM	
	CMD	**0x04** : WRITE CONTRAST Command Code
	VALUE	Contrast value: 0 to 15
	CHKSUM	Checksum byte
Direction	From **Host** to **Display**	
Length	Message length is **3** bytes	
Response	From **Display** to **Host**, ACK or NAK	
	ACK	If the Display understood the message, it will respond back to the host with an ACK after completing the requested action.
	NAK	If the Display did not understand the message it will respond with a NAK. In this case, the Host should retransmit the message.
Description	The host issues the **Write Contrast** command message when it wants to change the contrast or brightness of the display. Certain power saving modes and applications may require the host to dim or completely turn off the backlight. **Note:** Contrast value of 0 will turn the backlight OFF. Any non 0 value will turn the backlight ON.	
Example		

3.1.3.6. Report Object Status Message

Message	CMD, OBJ-ID, OBJ-INDEX, VALUE(msb), VALUE(lsb), CHKSUM	
	CMD	**0x05** : REPORT OBJECT Command Code
	OBJ-ID	Object ID. Refer to Object ID table for the relevant codes
	OBJ-INDEX	This byte specifies the index or the item number of the Object
	VALUE(msb)	Most significant byte of the 2 byte VALUE
	VALUE(lsb)	Least significant byte of the 2 byte VALUE
	CHKSUM	Checksum byte
Direction	From **Display** to **Host**	
Length	Message length is **6** bytes	
Response	No response required from **Host**.	
Description	This is the response message from the Display after the Host issues the **Read Object Status** message. The Display will respond back with the 2 byte value for the specific item of that object.	
Example		

3.1.3.7. Report Event Message

Message	CMD, OBJ-ID, OBJ-INDEX, VALUE(msb), VALUE(lsb), CHKSUM	
	CMD	**0x07** : REPORT EVENT Command Code
	OBJ-ID	Object ID. Refer to Object ID table for the relevant codes
	OBJ-INDEX	This byte specifies the index or the item number of the Object that caused the event
	VALUE(msb)	Most significant byte of the 2 byte VALUE
	VALUE(lsb)	Least significant byte of the 2 byte VALUE
	CHKSUM	Checksum byte
Direction	From **Display** to **Host**	
Length	Message length is **6** bytes	
Response	From **Host** to **Display**: NAK	
	NAK	If the Host did not understand the message it may respond with a NAK. In this case, the Display will retransmit the message.
Description	When designing the Genie display application in Workshop, each Object can be configured to report its status change without the host having to poll it (see *Read Object Status* message). If the object's 'Event Handler' is set to 'Report Event' in the 'Event' tab, the display will transmit the object's status upon any change. For example, Slider 3 object was set from 0 to 50 by the user.	
Example		

3.1.3.8. Write Magic Bytes

Message	CMD, STR-INDEX, LENGTH, "BYTES", CHKSUM	
	CMD	**0x08** : WRITE Magic Bytes Command Code
	OBJECT-INDEX	This byte specifies the index of the Magic Object
	LENGTH	Length of the array of bytes
	BYTES	An array of bytes
	CHKSUM	Checksum byte
Direction	From **Host** to **Display**	
Length	Message length is **4** bytes + the number of array bytes.	
Response	From **Display** to **Host**: ACK or NAK. The writing of this ACK is the responsibility of the Magic Object.	
	ACK	If the Display understood the message, it will respond back to the host with the ACK byte after completing the requested action.
	NAK	If the Display did not understand the message it will respond with the NAK byte. In this case, the Host should retransmit the message.
Description	This command can be used to send an array of bytes to a magic object. The magic object can process the bytes in any way you want it to as there is no restrictions on the format of the information sent. **Note1:** The maximum number of bytes that can be sent at once is set by the 'Maximum String Length' setting in Workshop4 under File, Options, Genie. **Note2:** A Workshop4 PRO license is required to use this capability.	
Example		

3.1.3.9. Write Magic Double Bytes

Message	CMD, STR-INDEX, LENGTH, "DOUBLEBYTES", CHKSUM	
	CMD	**0x09** : WRITE Magic Double Bytes Command Code
	OBJECT-INDEX	This byte specifies the index of the Magic Object
	LENGTH	Length of the array of Double bytes
	DOUBLEBYTES	An array of double bytes
	CHKSUM	Checksum byte
Direction	From **Host** to **Display**	
Length	Message length is **4** bytes + the number of array double bytes * 2.	
Response	From **Display** to **Host**: ACK or NAK. The writing of this ACK is the responsibility of the Magic Object	
	ACK	If the Display understood the message, it will respond back to the host with the ACK byte after completing the requested action.
	NAK	If the Display did not understand the message it will respond with the NAK byte. In this case, the Host should retransmit the message.
Description	This command can be used to send an array of Double bytes to a magic object. The magic object can process the double bytes in any way you want it to as there is no restrictions on the format of the information sent. **Note1:** The maximum number of double bytes that can be sent at once is set by the 'Maximum String Length' setting in Workshop4 under File, Options, Genie. The number is set inline with the guidelines for Unicode Strings. **Note2:** A Workshop4 PRO license is required to use this capability.	
Example		

3.1.3.10. Report Magic Bytes

Message	CMD, STR-INDEX, LENGTH, "BYTES", CHKSUM	
	CMD	**0x0A** : WRITE Magic Bytes Command Code
	OBJECT-INDEX	This byte specifies the index of the Magic Object
	LENGTH	Length of the array of bytes
	BYTES	An array of bytes
	CHKSUM	Checksum byte
Direction	From **Display** to **Host**	
Length	Message length is **4** bytes + the number of array bytes.	
Response	No response required from **Host.**	
Description	This command can be used to send an array of bytes to the host from a magic object. The magic object can send the bytes in any desired format. The Magic object is responsible for the complete building of this message. **Note:** A Workshop4 PRO license is required to use this capability.	

Example	seroutCS(REPORT_MAGIC_EVENT_BYTES) ; // we report in bytes
	seroutCS(object) ; // report object ID
	seroutCS(2) ; // two bytes following
	seroutCS(x) ; // write x as a byte
	seroutCS(y) ; // write y as a byte
	seroutOcs () ; // write out checksum	

3.1.3.11. Report Magic Double Bytes

Message	CMD, STR-INDEX, LENGTH, "DOUBLE BYTES", CHKSUM	
	CMD	**0x0B** : WRITE Magic Double Bytes Command Code
	OBJECT-INDEX	This byte specifies the index of the Magic Object
	LENGTH	Length of the array of double bytes
	DOUBLEBYTES	An array of double bytes
	CHKSUM	Checksum byte
Direction	From **Display** to **Host**	
Length	Message length is **4** bytes + the number of array double bytes * 2.	
Response	No response required from **Host.**	
Description	This command can be used to send an array of double bytes to the host from a magic object. The magic object can send the bytes in any desired format. The Magic object is responsible for the complete building of this message. **Note1:** A Workshop4 PRO license is required to use this capability.	

Example	seroutCS(REPORT_MAGIC_EVENT_DBYTES) ; // we report in bytes
	seroutCS(object) ; // report object ID
	seroutCS(2) ; // two Double bytes following
	seroutCS(x >> 8) ; // write x high byte
	seroutCS(x) ; // write x low byte
	seroutCS(y >> 8) ; // write y high byte
	seroutCS(y) ; // write y low byte
	seroutOcs () ; // write out checksum	

4. Objects Summary and Properties

This section provides a summary of all the objects along with some relevant information. For more detailed information on the Objects and their properties, refer to the individual application notes and the *ViSi-Genie User Guide*.

4.1. List of Objects

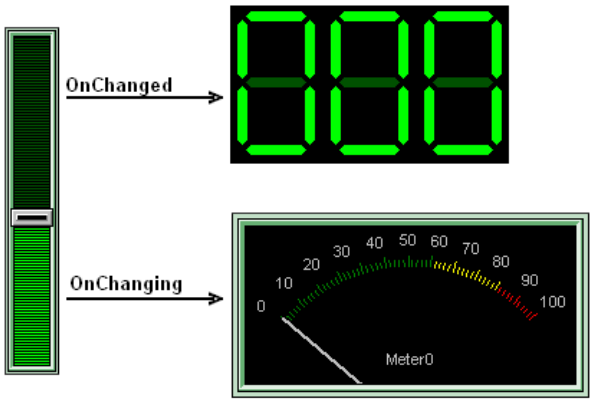
Legends used in this section:

- **On Actions:** These are the actions of an object that will influence other objects; OnChanged, OnChanging, OnActivate. These are selectable by the user under the 'Event' tab object properties in the Workshop4 Genie project.
- **OnChanged:** Other objects can be influenced when the state of the Object has changed.
- **OnChanging:** Other objects can be influenced whilst touch is maintained and the state of the object is changing.

4.2. Combining Objects

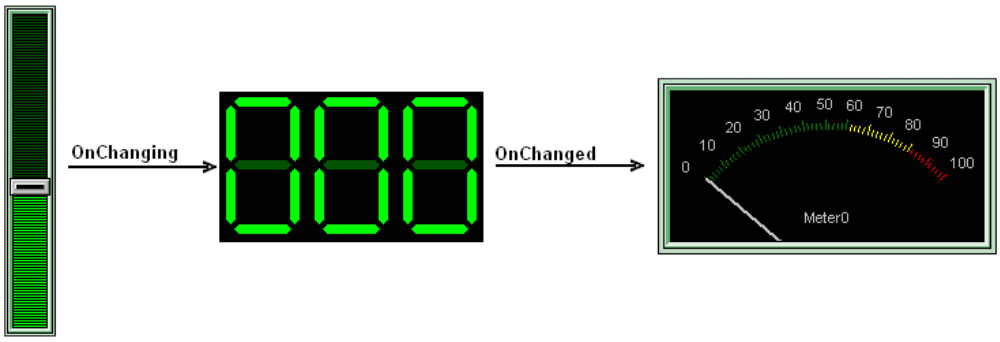
Combining the events with the objects allows multiple configurations.
The same track-bar object sends two different messages, each message being triggered by an event:

- The event onChanged sends a message to the LED digits;
- While the event onChanging sends a message to the meter.



Another configuration with a comparable result:

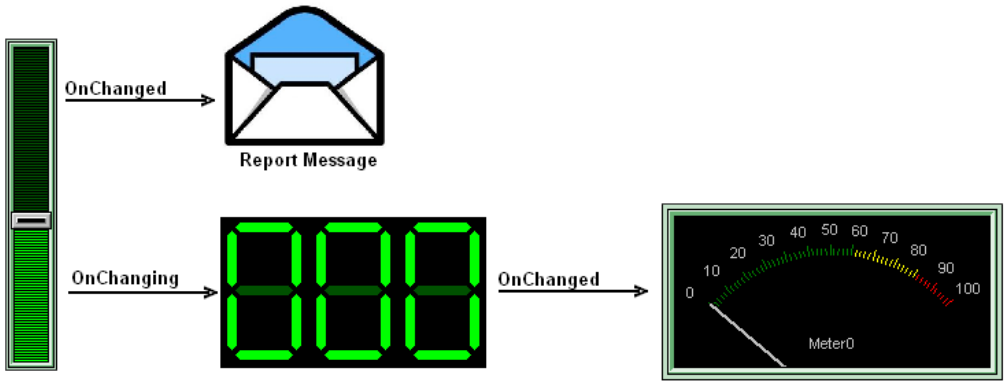
- Only one event is used, onChanging, and sends a message to a first object, the LED digit;
- The LED digit raises another event, onChanged, and sends a message to the second object, the meter.



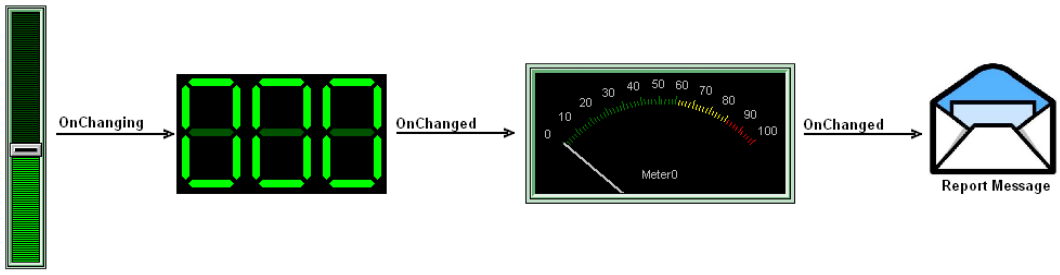
Now, a message can be sent the host controller, using the ReportMessage.

Below, the same track-bar object sends two different messages, each message being triggered by an event:

- The event onChanged sends a ReportMessage to the host controller;
- While the event onChanging sends a message to the LED digits;
- The LED digit raises another event, onChanged, and sends a message to the second object, the meter.



Another configuration with the same result: the objects are chained and the last one sends a ReportMessage to the host controller:



Note: (1) The visible properties of the objects are not applicable to the Genie application and can't be dynamically altered. The visible properties are adjustable and set only during the design phase under the 'Properties' tab in the Workshop4 Genie project.

Note: (2) To minimize the amount communications traffic during event reporting back to the host, it is advisable to select the 'OnChanged' event report option in the Genie project settings.

Note: (3) The host is able to alter the state of any Object by issuing the Write Object command message (with the exception of the Image object).

Note: (4) For the last combination, although the meter is set to report message back to the Host, the event reported would be actually that of the Slider which is the initiator of the event. A report is from the Input (ie a slider), not from any other widget in the chain.

4.2.1. Button Objects

4.2.1.1. Win Button

Object	Winbutton	
ID	6 (0x06)	
Description	A Windows Style Button Object. The button can be turned in to either a Momentary type or a Toggle type or a Matrix.	
Input	Yes	
Output	Yes	
Selected Properties	Matrix	Buttons can be matrixed (joined together). All buttons with the same Matrix number form one group. Only one member of a group can be down at any one time.
	Momentary	Specify Yes for a momentary Button, No for a toggle or matrixed Button in the off state, or On for a toggle or matrixed Button initialised in the on state. There is no check to ensure that only one button in the matrix is set to on, the button will just be set to on there is no 'reporting' of that state when it is set initially.
On Actions	OnChanged	When the Button is pressed and then released, the selection action occurs. This can cause a message to be sent to the host, or to activate another form or other actions to occur such as sounds, strings, timer or video objects.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the button is pressed and then released.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. For this object the Value(msb) is always **0x00.** The least significant byte, Value(lsb) will contain the Button state setting. **When reporting an Event or responding to query from Host:** For Toggle: If Button is OFF, when released: Value(lsb) = 0 If Button is ON, when released: Value(lsb) = 1 For Matrix: When Button is released: Value(lsb) = 1 For Momentary: When Button is released: Value(lsb) = 0

Note: (1) It is not recommended for the host to poll momentary type buttons as the Press/Release action can be missed. Instead, configure the display to automatically report the event.

Note: (2) If enabled as an option in the project tab, buttons can be disabled by writing 0xFFFF to them as their value. Buttons can be re-enabled by writing a 'normal', non 0xFFFF value to them

4.2.1.2. 4D Button

Object	4D Button	
ID	30 (0x1E)	
Description	A Generic Button Object for the various button styles available in Workshop. The 4D button can be turned in to either a Momentary type or a Toggle type or a Matrix.	
Input	Yes	
Output	Yes	
Selected Properties	Matrix	4D Buttons can be matrixed (joined together). All 4D buttons with the same Matrix number form one group. Only one member of a group can be down at any one time.
	Momentary	Specify Yes for a momentary 4D button, No for a toggle or matrixed 4d Button in the off state, or On for a toggle or matrixed 4d Button initialised in the on state. There is no check to ensure that only one button in the matrix is set to on, the button will just be set to on there is no 'reporting' of that state when it is set initially.
	Size	In general 4D buttons can be 32x3, 48x48, 64x64 or scaled. The scaled option allows you to scale the button to the exact size required. Depending upon the scaled size, this scaled button's quality may be compromised.
	Style	The style is generally the color of the button, but may represent something else.
	Type	Each different 4d button in the Object selector is a different type and that can be changed here, rather than deleting and re adding the 4D Button .
On Actions	OnChanged	When the 4D button is pressed and then released, the selection action occurs. This can cause a message to be sent to the host, or to activate another form or other actions to occur such as sounds, strings, timer or video objects.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the button is pressed and then released.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. For this object the Value(msb) is always **0x00.** The least significant byte, Value(lsb) will contain the 4D button state setting. **When reporting an Event or responding to query from Host:** For Toggle: If 4D Button is OFF, when released: Value(lsb) = 0 If 4D Button is ON, when released: Value(lsb) = 1 For Matrix: When 4D Button is released: Value(lsb) = 1 For Momentary: When 4D Button is released: Value(lsb) = 0
Notes	It is not recommended for the host to poll momentary type 4D buttons as the Press/Release action can be missed. Instead, configure the display to automatically report the event.	

Note: If enabled as an option in the project tab, 4D buttons can be disabled by writing 0xFFFF to them as their value. 4D buttons can be re-enabled by writing a 'normal', non 0xFFFF value to them.

4.2.1.3. Ani Button

Object	AniButton	
ID	31 (0x1F)	
Description	An Animated Button Object. The button can be turned in to either a Momentary type or a Toggle type or a Matrix.	
Input	Yes	
Output	Yes	
Selected Properties	Images	The list of images that make up the animated Button. Click on the ellipses to the right of the list to edit and maintain the Image list and display information about them.
	Interval	The interval (in ms) between the displays of successive images that make up the AniButton.
	Matrix	AniButtons can be matrixed (joined together). All Anibuttons with the same Matrix number form one group. Only one member of a group can be down at any one time.
	Momentary	Specify Yes for a momentary AniButton, No for a toggle or matrixed AniButton in the off state, or On for a toggle or matrixed AniButton initialised in the on state. There is no check to ensure that only one button in the matrix is set to on, the Anibutton will just be set to on there is no 'reporting' of that state when it is set initially.
	Stretch	Set stretch to Yes to automatically resize images to the size of the object.
On Actions	OnChanged	When the AniButton is pressed and then released, the selection action occurs. This can cause a message to be sent to the host, or to activate another form or other actions to occur such as sounds, strings, timer or video objects.
Event Report	OnChanged	Report Event message will be transmitted to the host after the button is pressed and then released.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. For this object the Value(msb) is always **0x00.** The least significant byte, Value(lsb) will contain the Button state setting. **When reporting an Event or responding to query from Host:** For Toggle: If AniButton is OFF, when released: Value(lsb) = 0 If AniButton is ON, when released: Value(lsb) = 1 For Matrix: When AniButton is released: Value(lsb) = 1 For Momentary: When AniButton is released: Value(lsb) = 0
Notes	Animated buttons consist of a sequence of images that are displayed in sequence to simulate the button going down (first to last image) and coming up (last to first image). In the unpressed state the first image is displayed, when the image is pressed (or touched) the second image is displayed, when the touch is released a timer is started and successive images are displayed until the last image is displayed. Sample images for a couple of buttons can be found in C:\Users\Public\Documents\4D Labs\Animated Buttons It is not recommended for the host to poll momentary type AniButtons as the Press/Release action can be missed. Instead, configure the display to automatically report the event.	

Note: If enabled as an option in the project tab, ANI buttons can be disabled by writing 0xFFFF to them as their value. ANI buttons can be re-enabled by writing a 'normal', non 0xFFFF value to them.

4.2.1.4. User Button

Object	User Button	
ID	33 (0x21)	
Description	A Generic Button Object for the user to create his or her own buttons with. The User button can be turned in to either a Momentary type or a Toggle type or a Matrix.	
Input	Yes	
Output	Yes	
Selected Properties	Images	The list of images that make up the User Button. Click on the ellipses to the right of the list to edit and maintain the Image list and display information about them.
	Matrix	User Buttons can be matrixed (joined together). All User buttons with the same Matrix number form one group. Only one member of a group can be down at any one time.
	Momentary	Specify Yes for a momentary User button, No for a toggle or matrixed 4d Button in the off state, or On for a toggle or matrixed User Button initialised in the on state. There is no check to ensure that only one button in the matrix is set to on, the button will just be set to on there is no 'reporting' of that state when it is set initially.
	Stretch	Set stretch to Yes to automatically resize images to the size of the object.
On Actions	OnChanged	When the User button is pressed and then released, the selection action occurs. This can cause a message to be sent to the host, or to activate another form or other actions to occur such as sounds, strings, timer or video objects.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the button is pressed and then released.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. For this object the Value(msb) is always **0x00.** The least significant byte, Value(lsb) will contain the 4D button state setting. **When reporting an Event or responding to query from Host:** For Toggle: If User Button is OFF, when released: Value(lsb) = 0 If User Button is ON, when released: Value(lsb) = 1 For Matrix: When User Button is released: Value(lsb) = 1 For Momentary: When User Button is released: Value(lsb) = 0
Notes	Buttons for 4D displays have 2 states when Momentary(Up and Pressed) and 4 states otherwise (Up, Up Pressed, Down and Down Pressed). User buttons can have more states, but these states can only be set by the host. **Note**: It is not recommended for the host to poll momentary type User buttons as the Press/Release action can be missed. Instead, configure the display to automatically report the event.	

Note: (1) User buttons can be 'Blocked'. This refers to the ability to have groups of buttons within the same button. Each group might be a different colour, have a different graphic, or have text in a different language. Please refer to the "ViSi-Genie Blocked User Buttons" application note for more information.

Note: (2) If enabled as an option in the project tab, User buttons can be disabled by writing 0xFFFF to them as their value. User buttons can be re-enabled by writing a 'normal', non 0xFFFF value to them.

4.2.2. Input Objects

4.2.2.1. Dip Switch

Object	Dipswitch	
ID	0 (0x00)	
Description	A Dip Switch Object that can have from 2 to 16 positions.	
Input	Yes	
Output	Yes	
On Actions	OnChanged	Other objects can be influenced when the switch position has changed, such as LED turns ON/OFF or 7segment display indicates position.
	OnChanging	Other objects can be influenced whilst touch is maintained and the switch position is changing. For example, if the Dip Switch has 16 positions, each intermediate change can dynamically display its position on a 7 segment display.
Event Report	OnChanged	**Report Event** message will be transmitted to the host once the position has changed and touch is released.
	OnChanging	**Report Event** message will be transmitted to the host as the position changes and whilst touch is maintained.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. For this object the Value(msb) is always **0x00.** The least significant byte, Value(lsb) will contain the dipswitch position settings. For 2 position Dip Switch: Value(lsb) = 0 or 1 For 3 position Dip Switch: Value(lsb) = 0 to 2 For N position Dip Switch: Value(lsb) = 0 to N-1 **Note:** N max = 16

4.2.2.2. Knob

Object	Knob	
ID	1 (0x01)	
Description	A Knob Object. The size, color and appearance of the knob are defined by the 'Backimage'. The size, color and appearance of the 'handle' (the red 'dot') are defined by the 'Handleimage'. These are adjustable under the object 'Properties' tab in the Genie project.	
Input	Yes	
Output	Yes	
On Actions	OnChanged	Other objects can be influenced when the knob position has changed. For example, the knob can be used as the frequency dial and the 7segment display indicates the new frequency when the change is made.
	OnChanging	Other objects can be influenced whilst touch is maintained and the knob position is changing. As per the example above, the 7 segment display can dynamically update the frequency values while the knob is rotated.
Event Report	OnChanged	**Report Event** message will be transmitted to the host once the position has changed and touch is released.
	OnChanging	**Report Event** message will be transmitted to the host as the position changes and whilst touch is maintained.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. The range of values for the knob can range from 0 to 65535 (0x00 to 0xFFFF). For example, if the knob value is 289 (0x0121) the 2 byte value will be: Value(msb) = 0x01 Value(lsb) = 0x21

4.2.2.3. Rocker Switch

Object	Rockerswitch	
ID	2 (0x02)	
Description	A Rocker Switch Object. This object has 2 positions, ON or OFF state.	
Input	Yes	
Output	Yes	
On Actions	OnChanged	Other objects can be influenced when the switch state has changed.
Event Report	OnChanged	**Report Event** message will be transmitted to the host once the switch position/state has changed and touch is released.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. For this object the Value(msb) is always **0x00.** The least significant byte, Value(lsb) will contain the switch position setting. For OFF state: Value(lsb) = 0 For ON state: Value(lsb) = 1

4.2.2.4. Rotary Switch

Object	Rotaryswitch	
ID	3 (0x03)	
Description	A Rotary Switch Object that can have from 2 to N positions.	
Input	Yes	
Output	Yes	
On Actions	OnChanged	Other objects can be influenced when the switch position has changed. For example a 7segment display indicates the switch position.
	OnChanging	Other objects can be influenced whilst touch is maintained and the switch position is changing. For example, if the Rocker Switch has 10 positions, each intermediate change can dynamically display its position on a 7 segment display.
Event Report	OnChanged	**Report Event** message will be transmitted to the host once the position has changed and touch is released.
	OnChanging	**Report Event** message will be transmitted to the host as the position changes and whilst touch is maintained.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. For this object the Value(msb) is always **0x00.** The least significant byte, Value(lsb) will contain the rocker switch position settings. For 2 position Rocker Switch: Value(lsb) = 0 or 1 For 3 position Rocker Switch: Value(lsb) = 0 to 2 For N position Rocker Switch: Value(lsb) = 0 to N-1 **Note:** Although there's no limit to the number of positions, for practical purposes limit N to 32.

4.2.2.5. Slider

Object	Slider	
ID	4 (0x04)	
Description	A Slider Switch Object.	
Input	Yes	
Output	Yes	
On Actions	OnChanged	Other objects can be influenced when the slider position has changed. For example, the slider can be used to set a volume level and the LED Bar Gauge can display the setting.
	OnChanging	Other objects can be influenced whilst touch is maintained and the slider position is changing. As per the example above, the LED Bar Gauge can dynamically update the level while the slider is moved.
Event Report	OnChanged	**Report Event** message will be transmitted to the host once the position has changed and touch is released.
	OnChanging	**Report Event** message will be transmitted to the host as the position changes and whilst touch is maintained.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. The range of values for the slider can range from 0 to 65535 (0x00 to 0xFFFF). For example, if the slider value is 50 (0x0032) the 2 byte value will be: Value(msb) = 0x00 Value(lsb) = 0x32

4.2.2.6. Trackbar

Object	Trackbar	
ID	5 (0x05)	
Description	The Trackbar Object.	
Input	Yes	
Output	Yes	
On Actions	OnChanged	Other objects can be influenced when the trackbar position has changed. For example, the trackbar can be used to set a volume level and the LED Bar Gauge can display the setting.
	OnChanging	Other objects can be influenced whilst touch is maintained and the trackbar position is changing. As per the example above, the LED Bar Gauge can dynamically update the level while the trackbar is moved.
Event Report	OnChanged	**Report Event** message will be transmitted to the host once the position has changed and touch is released.
	OnChanging	**Report Event** message will be transmitted to the host as the position changes and whilst touch is maintained.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. The range of values for the trackbar can range from 0 to 65535 (0x00 to 0xFFFF). For example, if the slider value is 300 (0x012C) the 2 byte value will be: Value(msb) = 0x01 Value(lsb) = 0x2C

4.2.2.7. Keyboard

Object	Keyboard	
ID	13 (0x0D)	
Description	A highly configurable Keyboard Object with 4 predefined configurations and an unlimited number of user definable configurations. Predefined configurations are: QWERTY, NUMERIC, CELLPHONE, CUSTOM	
Input	Yes	
Output	No	
On Actions	OnChanged	At the time of writing this document the keyboard object has no influence on other objects.
Event Report	OnChanged	**Report Event** message will be transmitted to the host along with the key value as soon as the key is pressed.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to the host. For the keyboard object the Value(msb) is always **0x00.** The least significant byte, Value(lsb) will contain the value of the key pressed. Value(lsb) = Key value
Notes	For more detailed information on the Keyboard objects and its, refer to the individual application notes and the *ViSi-Genie User Guide.*	

4.2.2.8. Color Picker

Object	Color Picker	
ID	32 (0x20)	
Description	The Color Picker Object.	
Input	Yes	
Output	No	
Selected Properties	Color	The background Color of the color Picker.
On Actions	OnChanged	The only meaningful Onchanged action is to create a report message.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the button is pressed and then released.
	Value(msb:lsb)	This is the 2 byte value that is used in the message transmissions to and from the host. For this object the Value represents the 16 bit color value in 565 format (5bits of Red, 6 bits of Green, 5 bits of Blue)**.**
Notes	The color of the color picker can be both read and set by the host.	

4.2.2.9. Smart Slider

Object	Smart Slider	
ID	36 (0x24)	
Description	A highly configurable Slider Object available in the **PRO** version of Workshop4	
Input	Yes	
Output	Yes	
On Actions	OnChanged	Other objects can be influenced when the slider position has changed. For example, the slider can be used to set a volume level and the LED Bar Gauge can display the setting.
	OnChanging	Other objects can be influenced whilst touch is maintained and the slider position is changing. As per the example above, the LED Bar Gauge can dynamically update the level while the slider is moved.
Event Report	OnChanged	**Report Event** message will be transmitted to the host once the position has changed and touch is released.
	OnChanging	**Report Event** message will be transmitted to the host as the position changes and whilst touch is maintained.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. The range of values for the slider can range from 0 to 65535 (0x00 to 0xFFFF). For example, if the slider value is 50 (0x0032) the 2 byte value will be: Value(msb) = 0x00 Value(lsb) = 0x32

4.2.2.10. Smart Knob

Object	Smart Knob	
ID	37 (0x25)	
Description	A highly configurable Knob Object available in the **PRO** version of Workshop4	
Input	Yes	
Output	Yes	
On Actions	OnChanged	Other objects can be influenced when the knob position has changed. For example, the knob can be used as the frequency dial and the 7segment display indicates the new frequency when the change is made.
	OnChanging	Other objects can be influenced whilst touch is maintained and the knob position is changing. As per the example above, the 7 segment display can dynamically update the frequency values while the knob is rotated.
Event Report	OnChanged	**Report Event** message will be transmitted to the host once the position has changed and touch is released.
	OnChanging	**Report Event** message will be transmitted to the host as the position changes and whilst touch is maintained.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. The range of values for the knob can range from 0 to 65535 (0x00 to 0xFFFF). For example, if the knob value is 289 (0x0121) the 2 byte value will be: Value(msb) = 0x01 Value(lsb) = 0x21

4.2.3. Gauge Objects

4.2.3.1. Angular Meter

Object	Angularmeter	
ID	7 (0x07)	
Description	An Angular Meter Object.	
Input	No	
Output	Yes	
On Actions	OnChanged	An input type object (such as a Slider, Trackbar, etc) can cause this output type Meter to be changed. This can subsequently cause another output type object (Digital Gauge, LED Digits, etc) to be changed.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the meter state has changed.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. The range of values for the meter (theoretically) can range from 0 to 65535 (0x00 to 0xFFFF). For example, if the meter value is 290 (0x0122) the 2 byte value will be: Value(msb) = 0x01 Value(lsb) = 0x22
See Also	Cool Gauge, Meter	

4.2.3.2. Cool Gauge

Object	Coolgauge	
ID	8 (0x08)	
Description	A Cool Gauge Object.	
Input	No	
Output	Yes	
On Actions	OnChanged	An input type object (such as a Slider, Trackbar, etc) can cause this output type Gauge to be changed. This can subsequently cause another output type object (Digital Gauge, LED Digits, etc) to be changed.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the gauge state has changed.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. The range of values for the gauge (theoretically) can range from 0 to 65535 (0x00 to 0xFFFF). For example, if the gauge value is 100 (0x0064) the 2 byte value will be: Value(msb) = 0x00 Value(lsb) = 0x64
See Also	Meter, Angular Meter	

4.2.3.3. Gauge (LED Type)

Object	Gauge	
ID	11 (0x0B)	
Description	A LED Type Gauge Object.	
Input	No	
Output	Yes	
On Actions	OnChanged	An input type object (such as a Slider, Trackbar, etc) can cause this output type Gauge to be changed. This can subsequently cause another output type object (Meter, LED Digits, etc) to be changed.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the meter state has changed.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. For example, if the Gauge value is 120 (0x0078) the 2 byte value will be: Value(msb) = 0x00 Value(lsb) = 0x78

4.2.3.4. Meter

Object	Meter	
ID	16 (0x10)	
Description	A Meter Object.	
Input	No	
Output	Yes	
On Actions	OnChanged	An input type object (such as a Slider, Trackbar, etc) can cause this output type Meter to be changed. This can subsequently cause another output type object (Digital Gauge, LED Digits, etc) to be changed.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the meter state has changed.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. The range of values for the meter (theoretically) can range from 0 to 65535 (0x00 to 0xFFFF). For example, if the meter value is 290 (0x0122) the 2 byte value will be: Value(msb) = 0x01 Value(lsb) = 0x22
See Also	Cool Gauge, Angular Meter	

4.2.3.5. Thermometer

Object	Thermometer	
ID	18 (0x12)	
Description	The Thermometer Object.	
Input	No	
Output	Yes	
On Actions	OnChanged	An input type object (such as a Slider, Trackbar, etc) can cause this output type Thermometer to be changed. This can subsequently cause another output type object (Meter, LED Digits, etc) to be changed.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the meter state has changed.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. For example, if the Thermometer value is 120 (0x0078) the 2 byte value will be: Value(msb) = 0x00 Value(lsb) = 0x78

4.2.3.6. Spectrum

Object	Spectrum	
ID	24 (0x18)	
Description	The Spectrum Object.	
Input	No	
Output	Yes	
Selected Properties	Bar Spacing	The number of unused pixels between each bar.
	Bar Width	The width of each bar.
	Columns	The number of bars.
	MinValue	Should always be 0 as there is no visual indication of value and all object values are 0 based.
	Spacing	The number of unused pixels between each tick on a bar.
	TickHeight	The height of each tick (row of color) on a bar.
	Width	The width is a function of BarSpacing, Barwidth, Bevel and colums, this value is ignored.
On Actions	N/A	
Event Report	N/A	
Notes	Values can only be meaningfully be written to a Spectrum whilst its form is displayed. Each value written to a spectrum is comprised of two bytes, the first byte is the bar (0 to Columns-1), the second byte is the value (0 to maxvalue). If the Form on which the Spectrum appears is changed all displayed values should be considered lost and must be resent from the host when the form containing the spectrum is redisplayed.	

4.2.3.7. Scope

Object	Scope	
ID	25 (0x19)	
Description	The Scope Object.	
Input	No	
Output	Yes	
Selected Properties	Color	The 'background' color of the scope.
	Graticule*	A dotted 'graticule' can be created with these properties.
	Refresh Increment	The scope is only redrawn after 'refresh increment' values (per trace) are written to it. This enables you to perform a 'mass' update without the redraw delay. An update can be 'forced' at any time by re-selecting the current form.
	Traces	Up to 4 traces can be drawn. Each value written to the object is written to the 'next' trace.
	Xmag	Used to compress or expand values in the X direction. Normally 0 for 'standard'. Be aware that negative values significantly increase the amount of internal memory required to hold scope values and that such values should be used carefully.
	Yamp	Amplification of values in the Y direction, 100 is unity and 200 is maximum.
	YLine*	A solid Line can be created across the scope at any point.
	Yoffset	The offset from the bottom of the scope of the 0 value. Normally 0 when drawing graphs or height / 2 otherwise.
On Actions	N/A	
Event Report	N/A	
Notes	Refer to the application notes for detailed information on the scope and its usage.	

4.2.3.8. Tank

Object	Tank	
ID	26 (0x1A)	
Description	The Tank Object.	
Input	No	
Output	Yes	
Selected Properties	Color	The color of the area surrounding the tank
	Container	The Tank can be thought of, roughly as a rectangular container with a shape on top of that. This defines that container and whether it is visible or not
	Min	Should always be 0 as there is no visual indication of value and all object values are 0 based.
	Position	The current fill position of the Tank, this is for design time visualization only, it does not have any effect at run time.
	Shape	The Shape at the top of the Tank.
On Actions	OnChanged	An input type object (such as a Slider, Trackbar, etc) can cause this output type Tank to be changed. This can subsequently cause another output type object (Meter, LED Digits, etc) to be changed.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the meter state has changed.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. For example, if the Tank value is 120 (0x0078) the 2 byte value will be: Value(msb) = 0x00 Value(lsb) = 0x78

4.2.3.9. Smart Gauge

Object	Smart Gauge	
ID	35 (0x23)	
Description	A highly configurable Gauge Object available in the **PRO** version of Workshop4. This can be in a form of a round gauge, a tank or thermometer or any output widget the user can think of.	
Input	No	
Output	Yes	
On Actions	OnChanged	An input type object (such as a Slider, Trackbar, etc) can cause this output type Gauge to be changed. This can subsequently cause another output type object (Digital Gauge, LED Digits, etc) to be changed.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the gauge state has changed.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. The range of values for the gauge (theoretically) can range from 0 to 65535 (0x00 to 0xFFFF). For example, if the gauge value is 100 (0x0064) the 2 byte value will be: Value(msb) = 0x00 Value(lsb) = 0x64
See Also	Meter, Angular Meter, Cool Gauge	

4.2.4. LEDs and Digits Object

4.2.4.1. LED

Object	Led	
ID	14 (0x0E)	
Description	The LED Object.	
Input	No	
Output	Yes	
On Actions	OnChanged	An input type object can cause this output type LED to be changed. This can subsequently cause another output type object to be changed.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the LED state has changed.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. For the LED object the Value(msb) is always **0x00.** The least significant byte, Value(lsb) will contain the state of the LED. If LED is OFF: Value(lsb) = 0 If LED is ON: Value(lsb) = 1
Notes	Glyph	If LedType is custom this Bitmap defines the Led that is displayed. The Bitmap should be two bitmaps side by side, the first being the 'OFF' image, the second being the 'ON' image.
	LedType	Can be set to three internal LED types or custom, in which case the LED is based on the Image contained in 'Glyph'.
See Also	User LED	

4.2.4.2. User LED

Object	Userled	
ID	19 (0x13)	
Description	The LED Object.	
Input	No	
Output	Yes	
On Actions	OnChanged	An input type object can cause this output type LED to be changed. This can subsequently cause another output type object to be changed.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the LED state has changed.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. For the LED object the Value(msb) is always **0x00.** The least significant byte, Value(lsb) will contain the state of the LED. If LED is OFF: Value(lsb) = 0 If LED is ON: Value(lsb) = 1
See Also	LED	

4.2.4.3. LED Digits

Object	Leddigits	
ID	15 (0x0F)	
Description	7 Segment LED Digits Object.	
Input	No	
Output	Yes	
On Actions	OnChanged	An input type object can cause this output type LED Digits to be changed. This can subsequently cause another output type object to be changed.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the LED Digits state has changed.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. For example, if the LED Digits value is 5645 (0x160D) the 2 byte value will be: Value(msb) = 0x16 Value(lsb) = 0x0D
See Also	Custom Digits	

Note: Leddigits normally display only positive integers. To support the usage of negative integers this option must be enabled on the Project Tab in Worksop. Remember that you may need to allow an extra digit if you intend displaying negative numbers.

4.2.4.4. Custom Digits

Object	Customdigits	
ID	9 (0x09)	
Description	The Custom Digits Object. The size, color and shape of the digits are defined by the 'Bitmap'.	
Input	No	
Output	Yes	
On Actions	OnChanged	An input type object (such as a Slider, Trackbar, etc) can cause this output type Digits to be changed. This can subsequently cause another output type object (Digital Gauge, LED Digits, etc) to be changed.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the digits state has changed.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. The range of values for the gauge (theoretically) can range from 0 to 65535 (0x00 to 0xFFFF). For example, if the digits value is 2100 (0x0834) the 2 byte value will be: Value(msb) = 0x08 Value(lsb) = 0x34
Notes	To create a custom bitmap, use GIMP, for example, type in the letters 0-9, adjust the fonts and attributes to obtain the desired appearance, then save the resulting image as a bitmap. The bitmap may need modifying, its width should be ten times the size of each digit. If using a project with negative integers and/or leading blank support you need to add two more characters to the bitmap, a blank and a minus sign.	
See Also	Led Digits	

Note: Customdigits normally display only positive integers with leading zeros. To support the usage of negative integers and/or leading blanks this option must be enabled on the Project Tab in Worksop. Remember that you may need to allow an extra digit if you intend displaying negative numbers.

4.2.5. Text and String Objects

4.2.5.1. Static Text

Object	Statictext
ID	21 (0x15)
Description	The Static Text Object.
Input	No
Output	No
On Actions	Not Applicable
Event Report	Not Applicable
Notes	Static Text is displayed as part of the form, there is no need to alter it.

4.2.5.2. Strings

Object	Strings	
ID	17 (0x11)	
Description	The Strings Object.	
Input	No	
Output	Yes	
On Actions	OnChanged	An input type object (such as a Button, Slider, Trackbar, etc) can cause this output type String to be changed. A string can be made up of many segments of messages (each separated by 0x0A Carriage Return). A button or other input type object can sequence thru these messages. Very handy when different messages need to be displayed upon certain actions taken. A state change in the string can subsequently cause another output type object to be changed.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the string state has changed.
	Value(msb:lsb)	Not used.
Notes	The first strings are displayed initially. Normally strings are set to predefined values, e.g. a value of 0 might display the string 'Hello There'. Using predefined values makes the most efficient use of the comms link and also minimizes the code required in your controller. In order to display a dynamically created string the user can send the *Write String ASCII* command message. The default maximum string length is 75, this can be changed in the Workshop4 options for Genie. For Unicode string objects Unicode strings can be sent, using the *Write String Unicode* command message. CRs and LFs can be included and the user is responsible for the 'formatting' of the string, the formatting of strings in Workshop4 does not apply to dynamic strings. Refer to the application notes for detailed information on Strings and their usage.	

4.2.6. System and Media Objects

4.2.6.1. Form

Object	Form	
ID	10 (0x0A)	
Description	A Form Object (a page on the screen).	
Input	No	
Output	Yes	
On Actions	OnActivate	An input type object (such as Button) can cause a form to be activated, along with all the objects on that form.
Event Report	OnActivate	**Report Event** message will be transmitted to the host after the form is activated.
	Value(msb:lsb)	Not used.
Notes	Form0 (or the first form) is automatically made active when the Genie application program starts on the display. The host can change the form by setting the value of the Form's index and sending the *Write Object Value* message. The selected form will then be displayed along with all of its objects and the ACK will be returned once this is complete.	

4.2.6.2. Image

Object	Image
ID	12 (0x0C)
Description	The Image Object.
Input	No
Output	No
On Actions	Not Applicable
Event Report	Not Applicable
Notes	Images are displayed as part of the form, there is no need to alter them.

4.2.6.3. Video

Object	Video	
ID	20 (0x14)	
Description	The Video Object.	
Input	No	
Output	Yes	
On Actions	OnChanged	An input type object, such as a button or a slider, can cause each frame of the video to be changed. This can subsequently cause another output type object to be changed, such as Led Digits as a frame counter.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the LED state has changed.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. This 2 byte field is the value of the video frame count.
Notes	**Note 1:** To use the video object as a video player, the Timer object must be used. Each click of the timer will increment to the next frame of the video. **Note 2:** The video object can be used as a slideshow. Compile all the separate images to a GIF file. A slider or a button can then be used to sequence thru the images as frames. **Note 3:** Refer to the video object application note for detailed information on the Video object and its usage.	

4.2.6.4. Sounds

Object	Sounds	
ID	22 (0x16)	
Description	The Sounds Object. The sound object can be made up of one or many wav files. Each wav file corresponds to an index within the sound object.	
Input	No	
Output	Yes	
On Actions	OnPlayingChanged OnVolumeChanged	When one of these values is changed by an input you can cause either another output to be changed or a message to be sent to the host.
Event Report	OnPlayingChanged OnVolumeChanged	**Report Event** message will be transmitted to the host after any of these states has changed.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. This 2 byte field hold the value of the specific action (see Notes below).
Notes	The Sound object is different to other objects in that there is only one of them (Sounds0) and that the values have predefined meanings, write to them to 'set' them. Reading Object # 0 returns the number of blocks left to play **Object Index** — **Meaning (Value field)** 0 — Play wav file n 1 — Set Volume n 2 — Pause 3 — Continue 4 — Stop An input can change the current wav file being played and/or change the volume of the sound object. For buttons this can be a discrete file, for other inputs care must be taken to ensure the value is valid and reasonable.	

Note: The Sound object (like the Timer object) will always reside in Form0.

4.2.6.5. Timer

Object	Timer	
ID	22 (0x17)	
Description	The Timer Object.	
Input	No	
Output	Yes	
On Actions	OnTimer	Normally used to move a video to the next frame.
Event Report	Not Applicable	
Notes	Enabled Set to yes to indicate the timer is to start when the Codeless program is loaded. Once enabled the timer continues until the Video displays its last frame. Interval The number of milliseconds between timer events.	

Note: The Timer object (like the Sounds object) will always reside in Form0.

4.2.6.6. User Images

Object	User Images	
ID	27 (0x1B)	
Description	The User Images Object represents an easy way to build a slideshow by joining together a sequence of images in one place.	
Input	No	
Output	Yes	
Selected Properties	Images	The list of images that make up the object. Click on the ellipses to the right of the list to edit and maintain the Image list and display information about them.
	Stretch	Set stretch to Yes to automatically resize images to the size of the object.
On Actions	OnChanged	An input type object, such as a button or a slider, can cause each frame of the User Images to be changed. This can subsequently cause another output type object to be changed, such as Led Digits as a frame counter.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after the User Images state has changed.
	Value(msb:lsb)	This is the 2 byte value, *Value(msb):Value(lsb),* that is used in the message transmissions to and from the host. This 2 byte field is the value of the User Images image count.
Notes	**Note 1:** To use the User Images object as a video player, the Timer object must be used. Each click of the timer will increment to the next frame of the User Images. **Note 2:** The User Images object can be used as a slideshow. A slider or a button can then be used to sequence thru the images as frames. **Note 3:** The User Images object operates in the same way as the Video object. Refer to the video object application note for detailed information on the Video object.	

4.2.7. Input/Output

4.2.7.1. Pin Output

Object	PinOutput	
ID	28 (0x1C)	
Description	The Pin Output Object.	
Input	No	
Output	Yes	
Properties	Idle State	A pin idle (i.e. be inactive) in either a low or high state. When the pin becomes active it will change to the opposite state.
	Pulse Duration	If 0 no pulse is generated and the pin moves to the selected state (active or idle). If non-zero (max 32000). a pulse to the active state is generated. This pulse is not exactly the specified pulse duration and may be slightly longer.
	Pin	The output pin to be pulsed. Multiple PinOutputs can use the same pin. It is the users' responsibility to manage such usage in a reasonable way.
On Actions	OnChanged	When one of these values is changed by an input you can cause either another output to be changed or a message to be sent to the host.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after any of these states has changed.
Notes	It would seem to make sense to only connect a momentary button to a pulsed output pin and similarly connected a toggled button to a non-pulsed output. Of course, it occasionally comes in handy to be able to do the non-apparent, so you can set these options any way you like. PinOutputs can be read by the host.	

Note: The PinOutput object (like the PinInput) will always reside in Form0.

4.2.7.2. Pin Input

Object	PinInput	
ID	29 (0x1D)	
Description	The Pin Input Object.	
Input	Yes	
Output	No	
Properties	Idle State	A pin idle (i.e. be inactive) in either a low or high state. When the pin becomes active it will change to the opposite state.
	Pin	The Input pin to be read. Multiple PinInputs can use the same pin. It is the users' responsibility to manage such usage in a reasonable way.
On Actions	OnChanged	When one of these values is changed you can cause either another output to be changed or a message to be sent to the host.
Event Report	OnChanged	**Report Event** message will be transmitted to the host after any of these states has changed.
Notes	Do not set a pin to both input and output as undesirable results may occur. If you need to read an output pin from your host then use the normal read command for the PinOutput.	

Note: The PinInput object (like the PinOutput) will always reside in Form0.

4.3. Object Summary Table

Object	ID	Input	Output	Notes
Dipswitch	0 (0x00)	✓	✓	
Knob	1 (0x01)	✓	✓	
Rockerswitch	2 (0x02)	✓	✓	
Rotaryswitch	3 (0x03)	✓	✓	
Slider	4 (0x04)	✓	✓	
Trackbar	5 (0x05)	✓	✓	
Winbutton	6 (0x06)	✓	✓	
Angularmeter	7 (0x07)		✓	
Coolgauge	8 (0x08)		✓	
Customdigits	9 (0x09)		✓	
Form	10 (0x0A)		✓	Used to set the current form
Gauge	11 (0x0B)		✓	
Image	12 (0x0C)			Displayed as part of form, no method to alter
Keyboard	13 (0x0D)	✓		Keyboard inputs are always single bytes and are unsolicited
Led	14 (0x0E)		✓	
Leddigits	15 (0x0F)		✓	
Meter	16 (0x10)		✓	
Strings	17 (0x11)		✓	
Thermometer	18 (0x12)		✓	
Userled	19 (0x13)		✓	
Video	20 (0x14)		✓	
Statictext	21 (0x15)			Displayed as part of form, no method to alter
Sound	22 (0x16)		✓	
Timer	23 (0x17)		✓	
Spectrum	24 (0x18)		✓	
Scope	25 (0x19)		✓	
Tank	26 (0x1A)		✓	
UserImages	27 (0x1B)		✓	
PinOutput	28 (0x1C)		✓	
PinInput	29 (0x1D)	✓		
4Dbutton	30 (0x1E)	✓	✓	
AniButton	31 (0x1F)	✓	✓	
ColorPicker	32 (0x20)	✓	✓	
UserButton	33 (0x21)	✓	✓	
MagicObject	34 (0x22)		✓	This object is only available with Workshop4 PRO
SmartGauge	35 (0x23)		✓	This object is only available with Workshop4 PRO
SmartSlider	36 (0x24)	✓	✓	This object is only available with Workshop4 PRO
SmartKnob	37 (0x25)	✓	✓	This object is only available with Workshop4 PRO

Note: Object IDs may change with future releases; it is not advisable to code their values as constants.

5. Workshop4 PRO

Workshop4 PRO is a licensed extension to Workshop4 to enable the usage of additional and specialised functionality.

Genie Magic - the ability to add standard 4DGL code at various points within the Genie environment - is the first addition to Workshop4 that requires a PRO license.

Another feature of Workshop4 PRO is the Smart Widgets Editor tool and the Smart Widgets. Smart Widgets Editor is a powerful utility that can be used to create complex widgets with up to six layers, including the layer for the base image. The layers can be arranged in any desired order. Each layer can contain one or more images arranged in sequence that can be manipulated in a variety of ways.

Image manipulation options include horizontal motion, vertical motion, and angular motion. Two or more layers can be linked to produce synchronized movements of images. This enables the generation of complex widgets with multiple moving parts.

Smart Widgets are custom widgets designed using the Smart Widgets Editor. There are three types available – Smart Gauges, Smart Knobs, and Smart Sliders.

When used in combination with Genie Magic, it is possible to use the Smart Widgets Editor tool and Smart Widgets to create other graphical user interface elements such as:

- dropdown menu
- side menu
- 'carousel slider'
- list box
- quarter-rotary menu
- dock menu
- fading slide-to-unlock button
- spectrum with transparency
- and many others

A seven day trial of Workshop4 PRO is available by clicking on the ‘Start PRO trial’ button under File, Options, License. This will enable use of PRO features for seven different, not necessarily consecutive, days.

A PRO license can be purchased from the 4D shopping cart, please include your Workshop4 serial number in the comments section when purchasing as this is required to generate your password to permanently unlock the functionality of Workshop4 PRO.

The email address used for the shopping cart becomes your registration email address. To have another address used as your registration email address, please also specify this email address in the comments section.

Your password will be emailed to the registration email address, this process is not fully automated as yet, so please allow at least one business day for the sending of your password.

The password and email registration should then be entered into the License screen and the ‘License PRO’ button should be pressed. Workshop4 will need to be restarted for the change to take effect.

6. Genie Magic

Genie Magic is an extension to Genie that requires Workshop4 PRO to function. It allows for the adding of any standard 4DGL code at various points in the Genie environment.

Extra functionality could include almost anything the user can imagine. For example:-

- Support R/W access to uSD from a host. (Example supplied)
- Produce sounds when objects are touched. (Example Supplied)
- Manipulate multiple objects in different ways by a single write command. (Example Supplied)
- Read data from I/O ports and write this data to Genie Objects. (Example Supplied)
- Collect and edit keyboard input sending result to host. (Example Supplied)
- Perform multiple updates with a single write command. (Example Supplied)
- 'Join' multiple objects together to get them to behave as a single object. (Example Supplied)
- Allow Genie to perform many more functions in a 'hostless' environment.

Code can be added at the following points

- Event Handlers
- Touch Press, move, release action 'points'
- Specific code positions
  - Constant / Global / Data definition
  - Main Loop
  - Pre/Post Activate Form
  - Pre/Post Genie Initialization
- An object to interface to the host

The host interface object can interface in the traditional Genie way using the normal Read/Write Object commands. It can also interface using 4 new commands which allow the sending and receiving of byte and double byte 'strings'. The format of these strings is very similar to the traditional 'write string' and 'write string Unicode' commands.

6.1. Genie Magic Callable Functions

This is a list of functions available within Genie that may be of use to the Genie Magic user. This list does not include all functions. This list includes functions 'generated' by Genie Magic.

Functions	Description	Parameters	Description
seroutCS	Writes parameter to Genie Serial port and updates output Checksum	Char	Char to write to port
seroutOcs	Writes Checksum to Genie Serial port and zeros output Checksum	None	
WriteObject	Sets the value of a Genie Object	ObjectType	Object Type, eg tKnob
		ObjectIdx	Object number, eg 0, 1
		NewVal	New value, eg 123
ActivateForm	Activates the for specified by the parameter	newform	The new form to be shown, eg 1
Timer*n*	Name format used by internally generated timer routines	None	
SendReport	Sends a standard 6 byte report Object or Event packet to the Genie Serial Port	Id	Report ID, eg REPORT_EVENT
		ObjectType	Object Type, eg tKnob
		ObjectIdx	Object number, eg 0, 1
		Value	Value, eg 123
seroutX	Defines serout for Genie Serial port	Char	Char to write to port
serinX	Defines serin for Genie Serial port	None	
rMagicObject*n*	MagicObject, used to receive data from a master	Action	The command that was received from the host, one of READ_OBJ, WRITE_OBJ, WRITE_MAGIC_BYTES or WRITE_MAGIC_DBYTES
		Object	Normal the object received from the host will be the same as the n in the function name, but since you could call this function internally it might be something else.
		newVal	N/A for Action of READ_OBJ, New value for Action of WRITE_OBJ, Otherwise number of parameters in the ptr array.
		ptr	N/A for Action of READ_OBJ and WRITE_OBJ, otherwise Pointer to array of parameters passed. Array is always a standard Picaso/Diablo integer array. For WRITE_MAGIC_BYTES each element contains a byte.
rMagicKbClrEvent*n*	Keyboard and Color Picker event	reportID	Report ID, always REPORT_EVENT
		objType	Object Type, eg tKeyboard
		objHash	Object number, eg 0, 1
		value	Value, eg 0x12

MagicEvent*n*	Magic Event	newval	The new value that is passed on to each event handler.
PrintStrings	Writes string objects	ID	String ID to write
		msgid	Message number for static strings, or pointer to word array being a Unicode string for dynamic strings.
		StringType	0 for static string, else dynamic string

6.2. Genie Magic Useful Variables

This is a list of global variables available within Genie that may be of use to the Genie Magic user. This list does not include all variables.

Variables	Description
ImageTouched	index of currently touched item
TouchXpos	X position of current touch
TouchYpos	Y position of current touch
CurrentForm	The number of the current form
hFonts	Array of handles for fonts, each element corresponds to the number of the relevant strings object
iSounds	Array of offsets into 'Sounds' Data of the filename associated with a given element
hndl	Handle for system image control

To access String object information, other than 'hFonts', access is required to the oStringss array variable. This variable is read only. oStrings[0] is the number of strings, ostringss[1] is a pointer to the strings0 information, if any of the pointers are 0xFFFF then this string is undefined (use the renumber tool to 'recover' this string number).

oStringss Offsets	Description
Ofs_String_x1	Left extent of strings object
Ofs_String_y1	Top extent of strings object
Ofs_String_x2	Right extent of strings object
Ofs_String_y2	Bottom extent of strings object
Ofs_String_FGColor	String Foreground Colour
Ofs_String_BGColor	String Background colour
Ofs_String_FontAttribs	Attributes for this strings object (Bold, Italic, Inverse, underlined). Formatted for passing to txt_Attributes() function.
Ofs_String_Transparent	1 Text will be drawn on solid rectangle of Ofs_String_BGColor. 0 Test will be drawn on form's background
Ofs_String_Form	Form on which this strings object normally appears
Ofs_String_StartH	High Offset into .txf file to start of this strings object's constant strings
Ofs_String_StartL	Low Offset into .txf file to start of this strings object's constant strings
Ofs_String_Size	Maximum length of constant string in .txf file
Ofs_String_Ansi	0 if constant strings in .txf file are Unicode, else ANSI

6.3. Genie Magic Useful Constants

This is a list of constants available within Genie that may be of use to the Genie Magic user. This list does not include all of constants.

Name	Value	Description
i*Objectn*	Varies	Index of object in Image Control. Eg iWinbutton0, iKnob6, etc.
ACK	0x06	Standard ASCII ACK code
NAK	0x15	Standard ASCII NAK code
tDipSwitch	0	Object index of the DipSwitch Object
tKnob	1	Object index of the tKnob Object
tRockerSwitch	2	Object index of the tRockerSwitch Object
tRotarySwitch	3	Object index of the tRotarySwitch Object
tGSlider	4	Object index of the tGSlider Object
tTrackbar	5	Object index of the tTrackbar Object
tWinButton	6	Object index of the tWinButton Object
tAngularmeter	7	Object index of the tAngularmeter Object
tCoolgauge	8	Object index of the tCoolgauge Object
tCustomdigits	9	Object index of the tCustomdigits Object
tForm	10	Object index of the tForm Object
tGauge	11	Object index of the tGauge Object
tImage	12	Object index of the tImage Object
tKeyboard	13	Object index of the tKeyboard Object
tLed	14	Object index of the tLed Object
tLeddigits	15	Object index of the tLeddigits Object
tMeter	16	Object index of the tMeter Object
tStrings	17	Object index of the tStrings Object
tThermometer	18	Object index of the tThermometer Object
tUserled	19	Object index of the tUserled Object
tVideo	20	Object index of the tVideo Object
tStaticText	21	Object index of the tStaticText Object
tSounds	22	Object index of the tSounds Object
tTimer	23	Object index of the tTimer Object
tSpectrum	24	Object index of the tSpectrum Object
tScope	25	Object index of the tScope Object
tTank	26	Object index of the tTank Object
tUserImages	27	Object index of the tUserImages Object
tPinOutput	28	Object index of the tPinOutput Object
tPinInput	29	Object index of the tPinInput Object
t4Dbutton	30	Object index of the t4Dbutton Object
tAniButton	31	Object index of the tAniButton Object
tColorPicker	32	Object index of the tColorPicker Object
tUserButton	33	Object index of the tUserButton Object
tMagicObject	34	Object index of the tMagicObject Object
tSmartGauge	35	Object index of the tSmartGauge Object
tSmartSlider	36	Object index of the tSmartSlider Object
tSmartKnob	37	Object index of the tSmartKnob Object

6.4. Genie Magic Objects

Event	'Location'	Useful Variables	Notes
Event	Called by the event handler of another object	newval	
Touch (press)	After img_Touched determination	ImageTouched TouchXpos TouchYpos	Set ImageTouched to -1 to stop Genie processing the object 'later on'. For example, this way you can turn a userbutton object into a multi position rocker, or a knob, etc.
(Touch) Move	After (Touch) Press	ImageTouched TouchXpos TouchYpos	Set ImageTouched to -1 to stop Genie processing the object 'later on'. For example, this way you can turn a userbutton object into a multi position rocker, or a knob, etc.
(Touch) Release	After (Touch) Move	ImageTouched TouchXpos TouchYpos	Set ImageTouched to -1 to stop Genie processing the object 'later on'. For example, this way you can turn a userbutton object into a multi position rocker, or a knob, etc.
KBClr	Called when a Key is pressed on the keyboard or when a colour is 'changed' in the event it is attached to.	value	
Code, Const/Global/Data	Outside of any function		
Code, MainLoop	Start of mainloop		
Code, PreActivateForm	Before old forms deactivation	newform CurrentForm	'CurrentForm' is the form that is 'currently' active. 'newform' is the form that is being activated. You could perform validation edits here and 'Return' to prevent the 'CurrentForm' being changed.
Code, PreGenieInit	At the very start.		
Code, PostActivateForm	After new form is activated	CurrentForm	
Code, PostGenieInit	After activation of the first form		

Obj		action	READ_OBJ, WRITE_OBJ, WRITE_MAGIC_BYTES or WRITE_MAGIC_DBYTES
		object	Usually the object number, but since you can call this routine inside your code you could change this.
		newVal	New value for WRITE_OBJ. Length for WRITE_MAGIC_BYTES and WRITE_MAGIC_DBYTES
		ptr	Pointer to data for WRITE_MAGIC_BYTES and WRITE_MAGIC_DBYTES, otherwise N/A

The Code insertion points can also be represented this way in a skeletal program:-

(CONSTANTS)
Constants/Global/Data
(Functions)

```
func ActivateForm()
	Pre Activate Form
	(Deactivate Old Form)
	(Display New Form)
	(Enable Inputs)
	Post Activate Form
endfunc

func main()
	Pre Genie Init
	(Mount SD)
	(Open Image Control)
	(Init 'constants')
	(Init Comms)
	(tag real objects)
	(Display Initial Form)
	Post Genie Init
	repeat
		Main Loop
	forever
endfunc
```

6.5. Genie Magic File Access Object

Magic File Access object is a Magic object written by 4D as a sample of what can be achieved using Genie Magic. This sample can be found in the ViSi Genie Magic samples folder.

The object can be used to access the micro SD card on the display. Files can be read and written to, as well as standard file manipulations. The object also has the ability to take screen copies.

The object is best understood by using the 'FileAccess' sample and using GTX to communicate with the object. GTX can also be used to communicate with this object if you implement it in your own programs. This can be achieved by copying the code contained in the MagicObject0(File access) object.

The object is written to using WRITE_MAGIC_BYTES, it reports back to the controlling program using REPORT_MAGIC_EVENT_BYTES.

These are the available functions:-

Function	Byte Value	Description and notes	Parameters	Response
MFILE_READ	0	Read a file. This reads the entire file.	Function code, Filename	More/final, bytes
MFILE_WRITE	1	Write to a file. The file must not currently exist. A maximum of xxx bytes can be written at once.	Function code, Filename, data	Null or True/False
MFILE_APPEND	2	Append to a file. The file must exist. A maximum of xxx bytes can be written at once.	Function code, Filename, data	Null, or True/False
MFILE_ERASE	3	Erase a file	Function code, Filename	True/False
MFILE_DIR	4	List files on the uSD card, filename can contain '*' wildcards at the end.	Function code, Filename	Bytes for each entry, Null at end
MFILE_SCREEN_CAPTURE	5	Capture the current screen on the display to a file. The file must not currently exist.	Function code, Filename	True/False
MFILE_SIZE	6	Report the size of a file	Function code, Filename	Null, or 4 bytes

Parameters (in addition to the parameters mandated by WRITE_MAGIC_BYTES)

Byte	What	Description
1	Function code	The function code
2-n	Filename	The null terminated 8.3 (Short) filename.
n+1…end	Data	Variable length data. Only used for file Write and Append functions

Please refer to the section Write Magic Bytes for more information on Parameters relating to the Writing of Magic Bytes.

Responses (in addition to the parameters mandated by REPORT_MAGIC_EVENT_BYTES)

Command(s)	Type	Notes
Read	Function code	MFILE_READ
	'Null'	Only the 'function code' will be sent if an error occurs (eg file does not exist)
	More / Final byte	0x00 indicates this is the last data for the file, 0xFF indicates another message follows this one.
	Bytes	Up to 255 bytes will be in each message. i.e. up to 253 bytes of file data
Write, Append	Function code	MFILE_WRITE or MFILE_APPEND
	'Null'	Only the 'function code' will be sent if file open fails
	1 byte	A single byte will be sent indicating the success(true, 1) or failure(false, 0) of the operation
Erase, Screen Capture	Function code	MFILE_ERASE or MFILE_SCREEN_CAPTURE
	1 byte	A single byte will be sent indicating the success(true, 1) or failure(false, 0) of the operation
Dir	Function code	MFILE_DIR
	Bytes	Each filename found will be returned in a single message. A 'Null' message containing only the 'function code' will be sent at the end.
Size	Function code	MFILE_SIZE
	'Null'	Only the 'function code' will be sent if an error occurs (eg file does not exist)
	4 Bytes	File size big Endian (i.e. most significant byte first)

Responses are in addition to the parameters specified in Section 3.1.2 in the "Command and Parameters Table" and 3.1.3.10 "Report Magic Bytes" sections. The responses above would fall into the 'Array' of data coming from the display module.

7. Revision History

Revision	Revision Content	Revision Date
1.0	First Release	19/11/2012
1.1	Fixed incorrect information in Section 2	28/02/2013
1.2	Updated description in 2.1.3.1	07/03/2013
1.3	Updated Note 4 in section 3.2	30/04/2013
1.4	Updated Buttons section 3.2.1.1	05/07/2013
1.5	Updated TOC	19/08/2013
1.6	Added New Widgets	18/09/2013
1.7	Updated strings object text in line with new Workshop4 options	21/03/2014
1.8	Updated Note 2 of Write String (ASCII) Message and Write String (Unicode) Message	05/06/2014
1.9	Added Genie PRO protocol information. Added Signed Led and Custom Digits information. Added information about disabled buttons	16/03/2015
1.10	Document PrintStrings and information relating to strings objects.	07/05/2015
1.11	Added Genie PRO object information. Added heading to describe graphical build process.	20/05/2015
2.0	Updated and changed formatting, added information on Smart Widgets and Smart Widgets Editor	24/7/2017
S 2.0	4D Systems release	24/07/2017
2.1, S 2.1	Updated Report Object section	13/04/2018
2.2	Updated formatting	05/04/2019
2.3	Fixed Typo in Footer	30/05/2019

8. Legal Notice

Proprietary Information

The information contained in this document is the property of 4D Labs Semiconductors and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission.

4D Labs Semiconductors endeavours 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 of 4D Labs Semiconductors products and services is continuous and published information may not be up to date. It is important to check the current position with 4D Labs Semiconductors. 4D Labs Semiconductors reserves the right to modify, update or makes changes to Specifications or written material without prior notice at any time.

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

Disclaimer of Warranties & Limitation of Liability

4D Labs Semiconductors makes no warranty, either expressed 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.

Images and graphics used throughout this document are for illustrative purposes only. All images and graphics used are possible to be displayed on the 4D Labs Semiconductors range of products, however the quality may vary.

In no event shall 4D Labs Semiconductors 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 4D Labs Semiconductors, or the use or inability to use the same, even if 4D Labs Semiconductors has been advised of the possibility of such damages.

4D Labs Semiconductors products are not fault tolerant nor designed, manufactured or intended for use or resale as on line control equipment in hazardous environments requiring fail – safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly to death, personal injury or severe physical or environmental damage (‘High Risk Activities’). 4D Labs Semiconductors and its suppliers specifically disclaim any expressed or implied warranty of fitness for High Risk Activities.

Use of 4D Labs Semiconductors’ products and devices in 'High Risk Activities' and in any other application is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless 4D Labs Semiconductors from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any 4D Labs Semiconductors intellectual property rights.

9. Contact Information

For Technical Support: www.4dlabs.com.au/support
For Sales Support: sales@4dlabs.com.au
Website: www.4dlabs.com.au