



The Big Book of SmartSet

A Designers' Guide to How It's S'posed to Work

Copyright© 2005 Elo TouchSystems.

All Rights Reserved

Content of this manual is proprietary to Elo TouchSystems, Inc. It may be shared with customers or vendors only under terms of a confidential non-disclosure agreement (CNDA).

Disclaimer

The information in this document is subject to change without notice. Elo TouchSystems makes no representations or warranties with respect to the contents hereof, and specifically disclaims any implied warranties of merchantability or fitness for a particular purpose. Elo TouchSystems reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Elo TouchSystems to notify any person of such revisions or changes.

Trademark

iTouch, IntelliTouch, SecureTouch, AccuTouch, Entuitive, MonitorMouse, ELODEV and SmartSet are trademarks of Elo TouchSystems.

Other product names mentioned herein may be trademarks or registered trademarks of their respective companies. Elo TouchSystems claims no interest in trademarks other than its own.

Table of Contents

Table of Contents	3
List of Tables	5
List of Figures	6
<i>How to use this document template</i>	7
Please read this before you make changes to the document	7
Document properties	7
Chapters	7
Headings and the Table of Contents	7
Tables, figures and captions	8
Appendices	8
Glossary, Bibliography and the Index	8
<i>Chapter 1: Touchscreens</i>	9
AccuTouch	9
IntelliTouch	12
Surface Capacitive	14
Four-wire Resistive	15
<i>Chapter 2: Coordinates</i>	16
Coordinate orientation	16
Elo “classic”	16
USB	17
Raw coordinates	18
Calibration	18
Scaling	18
Linearization	18
Edge acceleration	18
<i>Chapter 3: SmartSet commands</i>	19
Protocol layer	19
Message structure	19
List of Messages	22
Acknowledgement: ‘a’, ‘A’	22
Report timing: ‘B’, ‘b’	24
Calibration: ‘C’, ‘c’	25
Setting by range versus setting by parameter	27
Z-Axis Calibration	27
Diagnostics: ‘D’, ‘d’	28
Emulate: ‘E’, ‘e’	31
Filter Command – ‘F’, ‘f’	33

Configuration - 'g' _____	37
User Timer - 'H','h' _____	38
ID Command - 'i' _____	39
Jumpers Command - 'j' _____	42
Key Command - 'K','k' _____	44
Low Power - 'L','l' _____	45
Mode Command - 'M','m' _____	46
Nonvolatile RAM Command - 'N' _____	50
Owner Command - 'o' _____	51
Parameter Command - 'P','p' _____	52
Quiet Command - 'Q','q' _____	54
Reset Command - 'R' _____	55
Scaling Command - 'S','s' _____	56
Touch Command - 't' _____	59
Upload Command - 'U','u' _____	60
Subcommand 'S' - Query size of block _____	61
Subcommand 'C' - Query content of block. _____	61
IntelliTouch upload block content _____	62
Carroll Touch upload block content _____	64
Subcommand 'M' - Set or query mode _____	67
Subcommands 'W' and 'N' - Set or query Serial Number _____	68
Subcommand 'U' - Debug monitor commands. _____	71
Chapter 4: RS-232 serial communication _____	72
Signals, timing, character codes, etc. _____	72
SmartSet adaptation to RS-232 _____	72
Chapter 5: USB communication _____	74
Signals, tokens, packets, etc. _____	74
SmartSet adaptation to USB _____	74
Appendix A: (style ChapterTitle) _____	75
Heading 2 _____	75
Heading 3 _____	75
Heading 4 _____	75
Heading 5 _____	75
Glossary (style GlossaryTitle) _____	76
Bibliography (style BiblioTitle) _____	77
Heading 2 _____	77
Heading 3 _____	77
Heading 4 _____	77
Heading 5 _____	77
Index _____	78

List of Tables

Table 1 A typical table	8
Table 2 Prefixes used in Hungarian Notation.....	20
Table 3 Error codes	22
Table 4 Terms and their definitions.....	76

List of Figures

Figure 1 A shadowy figure	8
Figure 2 AccuTouch touchscreen	10
Figure 3 AccuTouch "classic" connector	10
Figure 4 AccuTouch "rational" connector	10
Figure 5 Effect of inverting AT connector	11
Figure 6 IntelliTouch touchscreen	12
Figure 7 IntelliTouch "classic" connector	13
Figure 8 IntelliTouch "Flipper" connector.....	13
Figure 9 Surface capacitive touchscreen	14
Figure 10 Standard surface capacitive connector	14
Figure 11 Four-wire resistive touchscreen	15
Figure 12 Four-wire resistive connector	15
Figure 13 X,Y axis orientation in RS232	16
Figure 14 X, Y axis orientation in USB	17

How to use this document template

Please read this before you make changes to the document

The manual template operates through Word's "section" and "style" features. I am sick to death of documents that have a million paragraphs of Normal style, each with a slightly different typography. This template uses different styles to vary the appearance of paragraphs.

If you need to change a paragraph's font, borders or other attributes, consider adding a new style.

Document properties

Use the menu entry "File" >> "Properties" to set the document title, subtitle and other values that are inserted as field codes. Set the title and subtitle are from the properties "General" tab. Set the document number, designation (part number or document number, for example) and revision level on the "Custom" tab.

Chapters

Each chapter title begins with the word "Chapter" followed by the chapter number. Numbering is automatic using a Word "Seq" field code.

Bowing to Microsoft's total inflexibility, I found it necessary to set the ChapterTitle style to be the same thing as Heading 1. This makes the automatically generated table of contents come out looking right.

The simplest way to add a new chapter is just to copy one of the existing chapter titles. When you do this, position the cursor at the start of the chapter title and select the menu item "Insert" >> "Break..." >> "Next page". Inserting a section break this way allows you to control the header, footer and page layout formatting for the chapter independently from the rest of the document.

Headings and the Table of Contents

Word automatically generates the table of contents from paragraphs with Heading styles. Apply the ChapterTitle style to each chapter title. For subheadings and lower level outlining, use Heading 2, Heading 3 and so on.

Because Heading 1 functions the same as the ChapterTitle style, you can use Heading 1 for titling unnumbered sections like the preface and introduction.

Tables, figures and captions

Use the menu entry “Table” to insert and format tables.

Mark each table with a caption. Position the cursor on the line prior to the table, and select the menu entry “Insert” >> “Reference” >> “Caption”. You can select a label of type “figure”, “table” or “equation”. Word numbers the tables from these captions, and it generates the list of tables automatically.

Table 1 A typical table

Column Headings in style NormalBoldCentered		
Column entries in style Normal		

Number the figures using the same technique. After you insert a picture or graph, select it and apply the paragraph style “Figure”.



Figure 1 A shadowy figure

Appendices

Appendix titles are formatted with the paragraph style ChapterTitle. However, appendices are automatically numbered with capital letters. The letters are generated using an independent series of “Seq” field codes.

Glossary, Bibliography and the Index

The paragraph styles GlossaryTitle, BiblioTitle and IndexTitle are derived from Heading 1 to make the Table of Contents come out looking right. These styles are the same as the ChapterTitle style, but they don’t reserve large blocks of following white space.

Another bow to Microsoft: a string of empty paragraphs here makes it possible to access all three sets of section headers and footers – first page, odd page and even page.

Chapter 1: Touchscreens

AccuTouch

AccuTouch is Elo's five-wire resistive touch sensor technology. The standard orientation for an AccuTouch touchscreen is shown in the figure below.

During normal operation the controller applies three/four distinct drive conditions to the touchscreen.

- X-coordinate sensing: The controller drives substrate corners X and H high, and it drives corners Y and L low. If there is a touch, then the voltage on the "sense" terminal represents the touch's left-to-right location.
- Y-coordinate sensing: The controller drives substrate corners Y and H high, and it drives corners X and L low. If there is a touch, then the voltage on the "sense" terminal represents the touch's bottom-to-top location.
- Touch detection: All substrate corners are driven high, and the sense terminal is connected through a high resistance (10 k-Ohms to 20 k-Ohms) to a ground. If there is no touch, then the sense terminal voltage is near ground. If there is a touch, then the sense voltage terminal is high.
- **COACH3 Touch detection: When no touch is applied to the touchscreen, the COACH3 chip sits in a polling loop waiting to detect touch activity. In this state, all four substrate electrodes are pulled high internally in the chip and the "SENSE" line is at ground. The measurement system, "TOUCH" connected to "Y", detects a high with no touch. When a touch occurs, the "TOUCH" lead is measured at a low. This condition causes the coordinate measurement process to begin. (JASON FORD)**

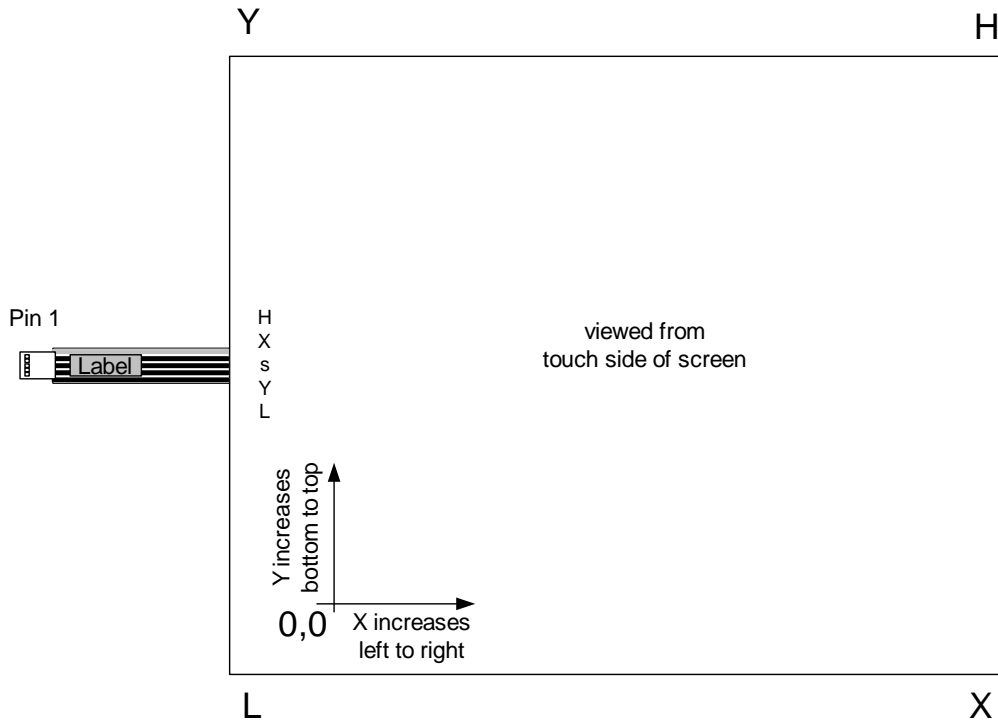


Figure 2 AccuTouch touchscreen

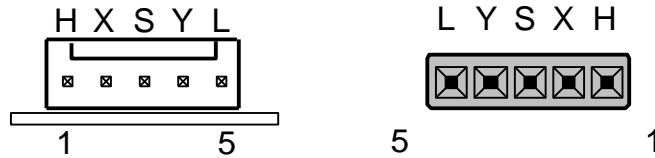


Figure 3 AccuTouch "classic" connector

The classic AccuTouch pin configuration solved compatibility issues when 5-wire technology displaced Elo's earlier 3-wire diode touchscreens. A controller designed to operate AccuTouch 5-wire touchscreens also operates 3-wire touchscreens.

Modern screen production techniques require a "crossover" in the wiring, either in the flex cable or on the screen itself. The rational pin configuration eliminates the need for a crossover, reducing the cost of the screen.

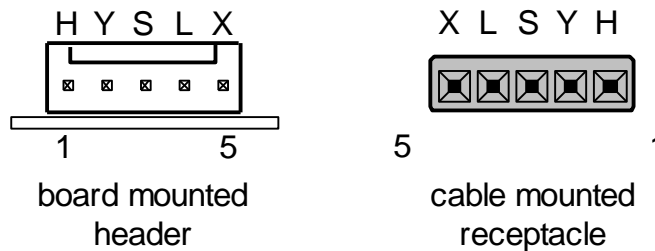


Figure 4 AccuTouch "rational" connector

Most AccuTouch touchscreens are built with a receptacle that is not polarized. The receptacle can be plugged into the controller's header upside down. Because the sense lead is located on the connectors' center pins, the controller and touchscreen work when the connector is inverted. However, the coordinate axis orientation changes. The classic and rational connector styles produce different axis orientations. A controller in the classic configuration does not work with a rational touchscreen.

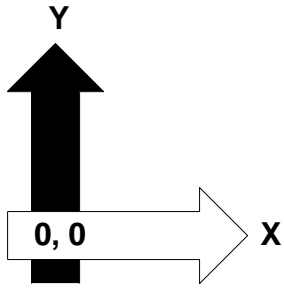
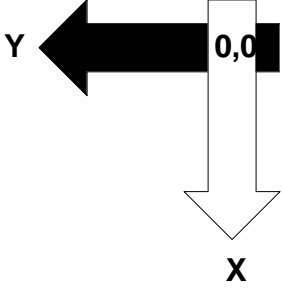
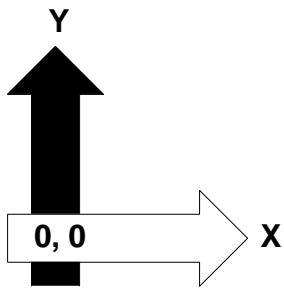
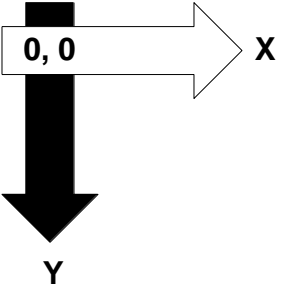
Controller expects classic AT.	Actual is classic AT plugged upside down.
1 = H 2 = X 3 = sense 4 = Y 5 = L	1 = L 2 = Y 3 = sense 4 = X 5 = H
	
Controller expects rational AT.	Actual is rational AT plugged upside down.
1 = H 2 = Y 3 = sense 4 = L 5 = X	1 = X 2 = L 3 = sense 4 = Y 5 = H
	

Figure 5 Effect of inverting AT connector

IntelliTouch

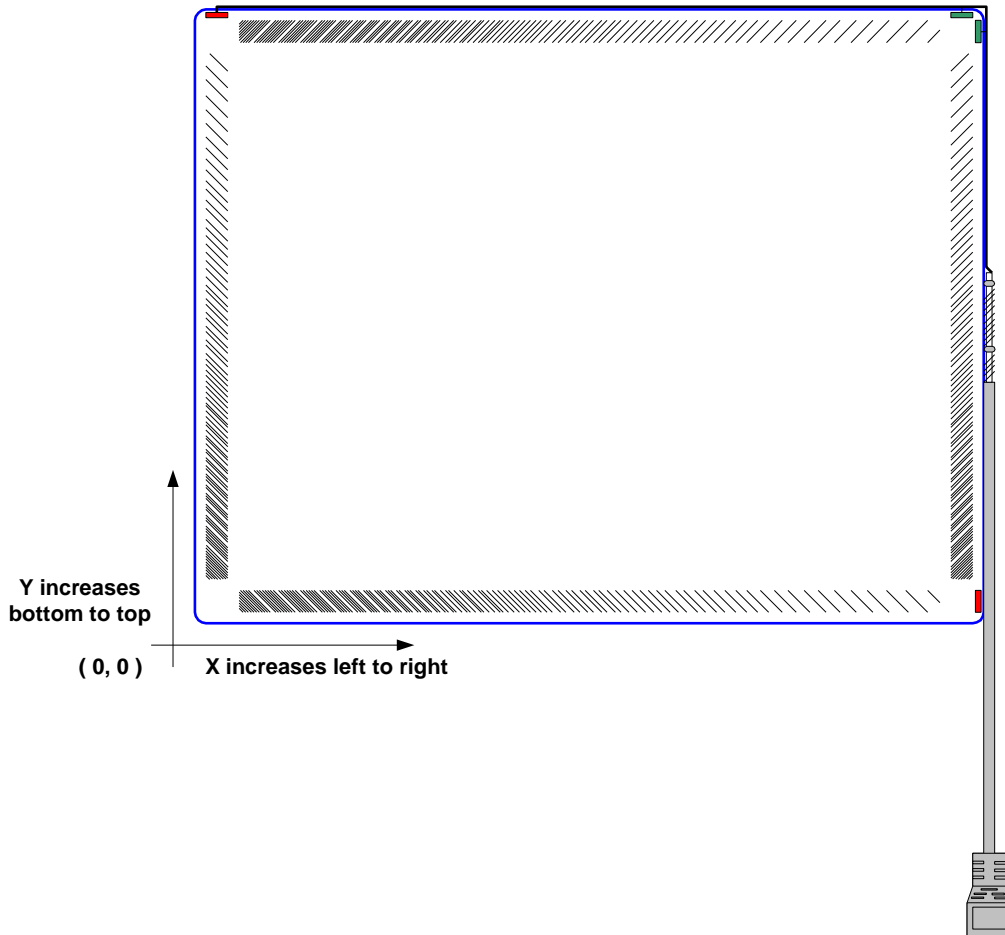


Figure 6 IntelliTouch touchscreen

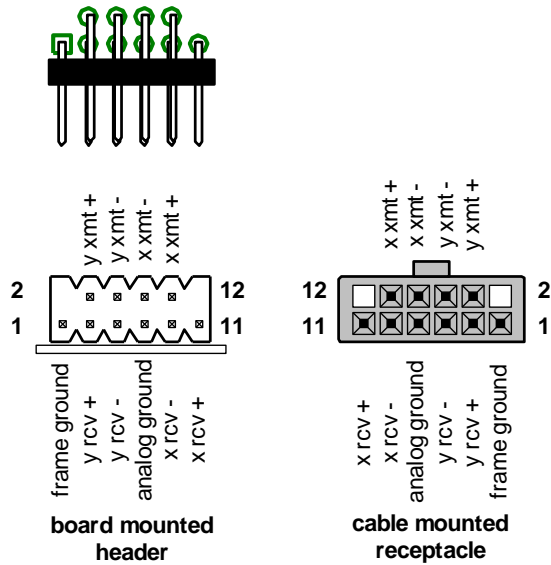


Figure 7 IntelliTouch "classic" connector

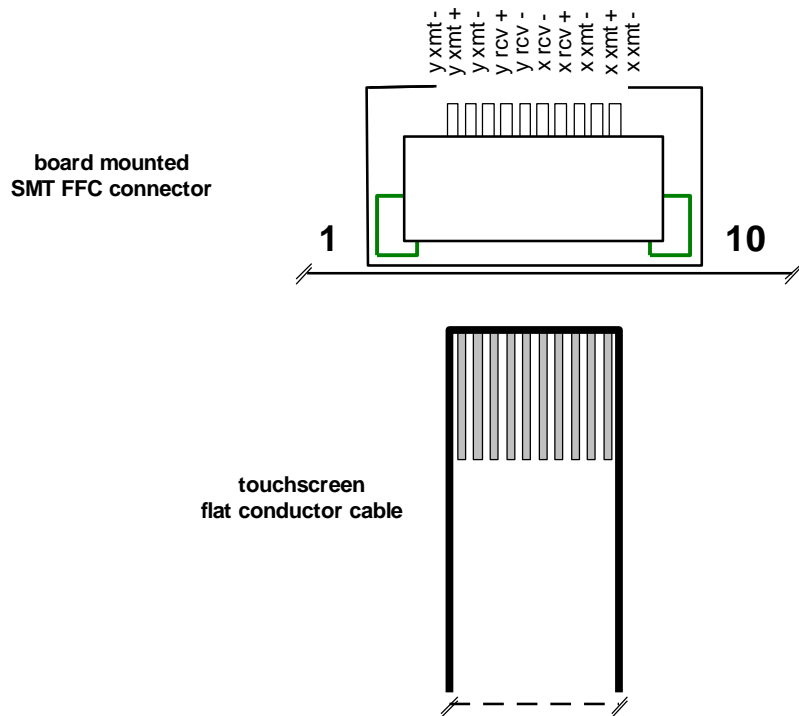


Figure 8 IntelliTouch "Flipper" connector

Surface Capacitive

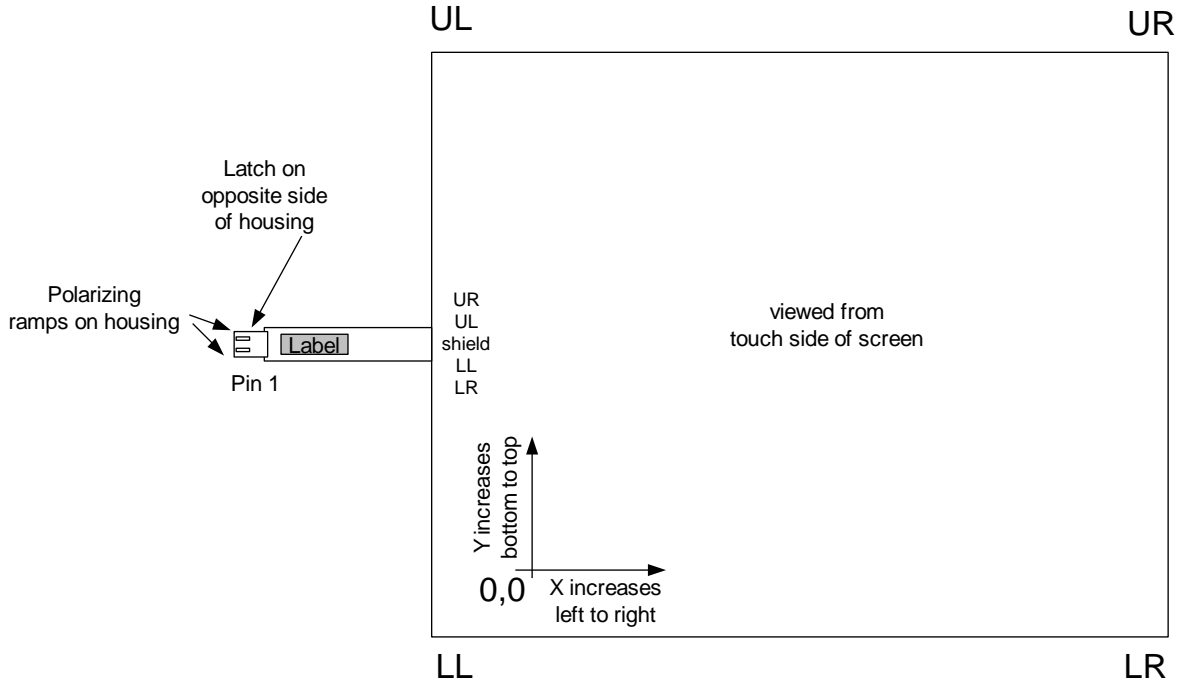


Figure 9 Surface capacitive touchscreen

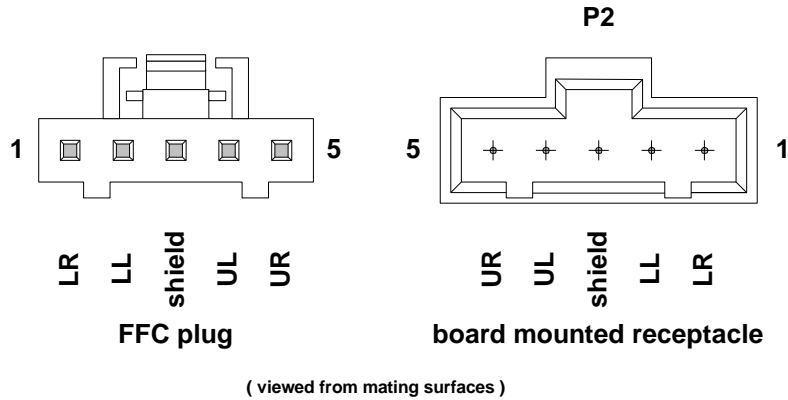


Figure 10 Standard surface capacitive connector

Four-wire Resistive

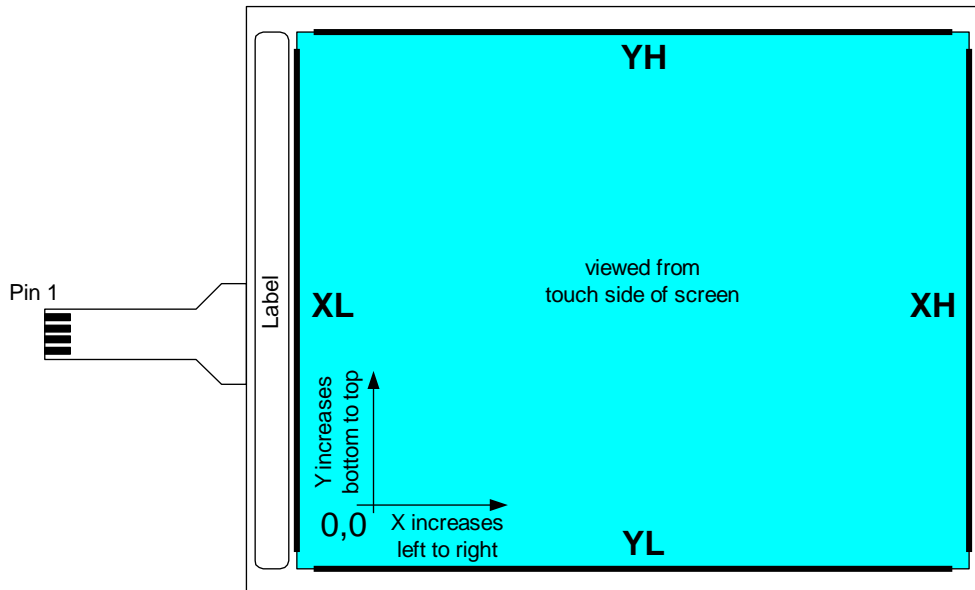


Figure 11 Four-wire resistive touchscreen

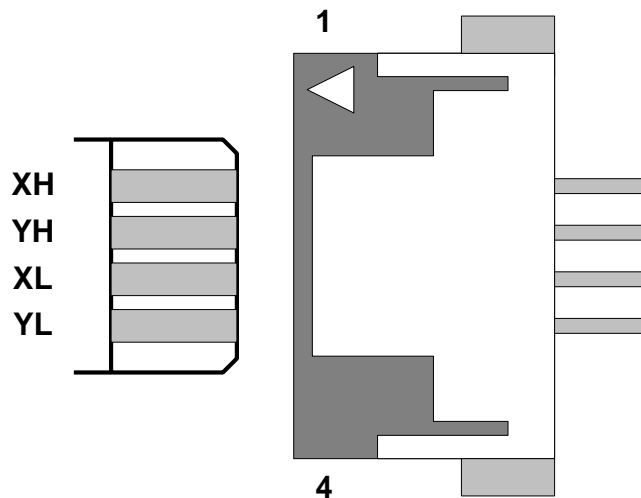


Figure 12 Four-wire resistive connector

Chapter 2: Coordinates

Coordinate orientation

Elo “classic”

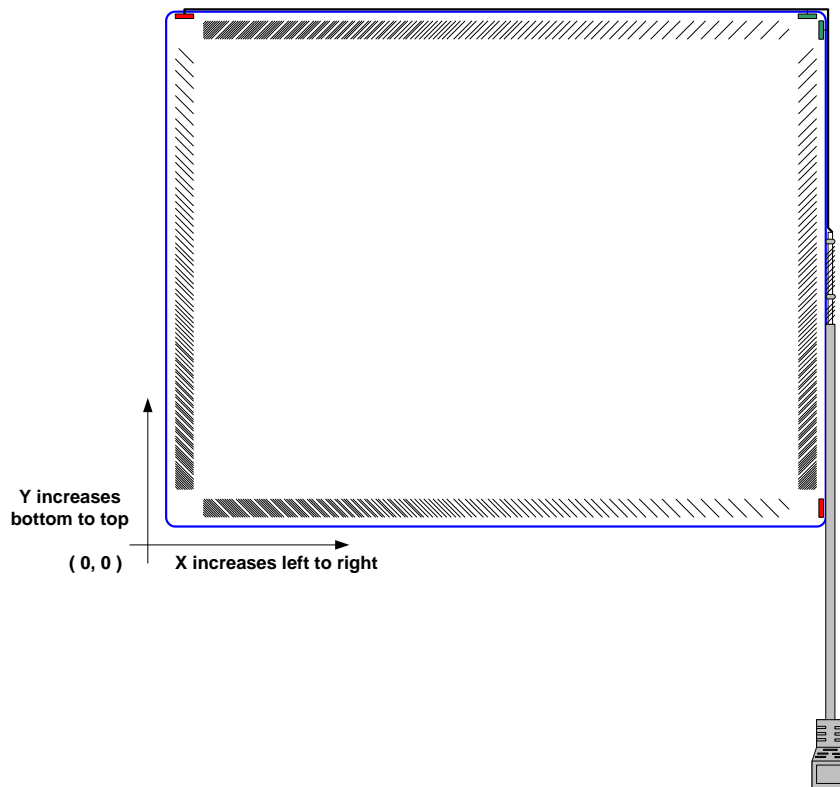


Figure 13 X,Y axis orientation in RS232

When the controller is connected to a touchscreen with normal wiring, the origin for the coordinate system is located at the bottom left corner of the touch surface. This orientation applies when the controller is communicating via an RS-232 serial port, when controller calibration and scaling features are disabled and when the touchscreen is positioned as shown in the figure above. This is consistent with Elo AccuTouch and Surface Capacitive touchscreens.

USB

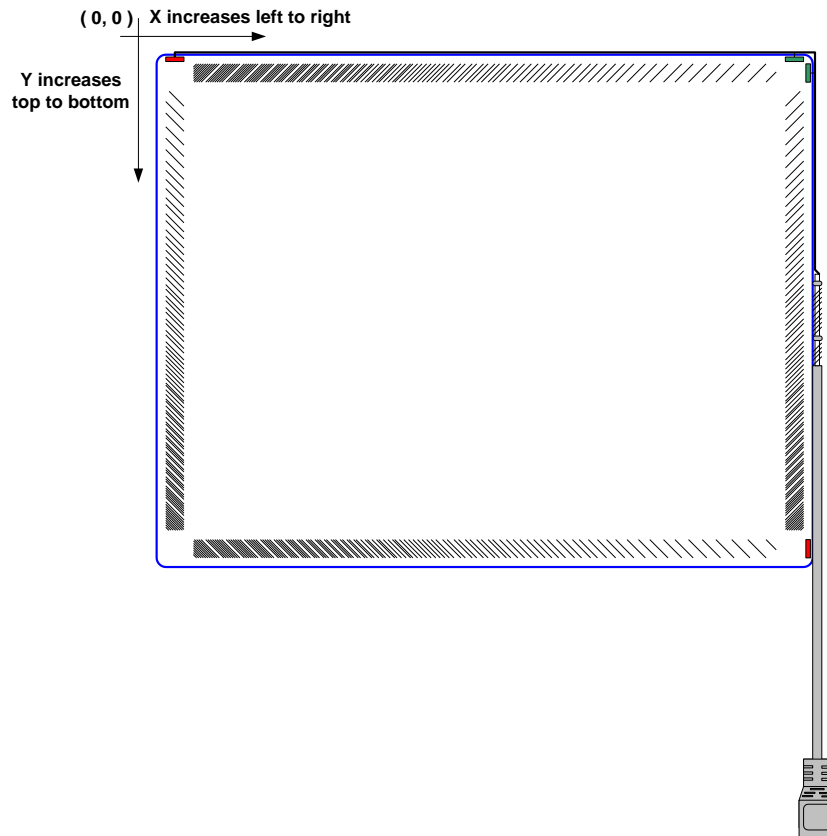


Figure 14 X, Y axis orientation in USB

When the controller is connected to a touchscreen with normal wiring, the origin for the coordinate system is located at the top left corner of the touch surface. This orientation applies when the controller is communicating via USB, when controller calibration and scaling features are disabled and when the touchscreen is positioned as shown in the figure above. This orientation complies with section 5.9, "Orientation", of the USB Device Class Definition for Human Interface Devices (HID).

Raw coordinates
Calibration
Scaling
Linearization
Edge acceleration

Chapter 3: SmartSet commands

“SmartSet” is Elo’s host/controller communication format. The core of SmartSet is a collection of eight-byte commands. Virtually any two-way communication channel can be adapted to carry SmartSet formatted traffic. This chapter describes the top-level protocol for exchanging SmartSet commands, and it gives a listing of the commands.

Protocol layer

SmartSet is light on protocol because its original design concentrated more on individual commands than on system-level concerns.

When the controller finishes processing a command, it sends an “acknowledgement” response. This is true even if the controller sends other responses as part of processing the command. If the controller encountered any errors during command processing, it reports them in the concluding acknowledgement response. Unfortunately, there are a couple of exceptions to this rule.

- A “cold boot” reset command does not conclude with an acknowledgement response.
- A “quiet” command that turns off acknowledge reporting does not conclude with an acknowledgement response.

After it sends a command, the host must wait for an acknowledgement before sending another command. Although some controllers tolerate two or three successive commands, the host shouldn’t expect a controller to queue up multiple commands.

If the command and response interchange loses synchronization, the situation is undefined. Use error recovery resources available through the underlying communication channel.

Message structure

A SmartSet message is eight bytes long. The communication channel can carry the 8-byte message structure as an 8-character string, an array of four words, an array of two double words or any other convenient form.

Message structures are described in tables that appear throughout this chapter. For convenience, message fields are named using Hungarian notation. In this notation, lower case prefixes provide clues to size and usage of the named field. The table below lists some prefixes that occur in this notation.

Table 2 Prefixes used in Hungarian Notation

Prefix	Meaning
“a”	Byte interpreted as an ASCII character code
“b”	Byte interpreted as an unsigned 8-bit integer
“bm”	Bit-mapped byte
“f”	Flag, a single-bit variable
“ff”	A bit field that occupies less than a full byte
“w”	Word, a two-byte entry interpreted as an unsigned 16-bit integer. The least significant byte occurs at the lowest address or first in time (little-endian order).
“i”	Integer, a two-byte entry interpreted as a signed 16-bit integer. The least significant byte occurs at the lowest address or first in time (little-endian order).
“dw”	Double word, a four-byte entry interpreted as an unsigned 32-bit integer. The least significant byte occurs at the lowest address or first in time (little-endian order).

The general form for a SmartSet message is this.

“Message name” “Message function”			
Offset	Length	Name	Usage
0	1	aTag	Fixed value is given here.
1	7		Additional fields are optional.

“Message name” is a convenient name used to refer to the command in text descriptions.

“Message function” is one of these: “command”, “query” or “response”. Read the discussion of the field “aTag” below.

Entries in the “Offset” column give the distance from the start of the message to the start of each field in bytes.

Entries in the “Length” column give the length of each field in bytes.

Entries in the “Name” column give the Hungarian notation identifier for each field.

Entries in the “Usage” column describe the field.

The field named “aTag”, the first byte in the message, defines the type of message. Content of this field is an ASCII printable character. The message function and the direction of communication affect the value of this tag.

- If the host sends a message that tells the controller to change its configuration or to perform an operation, then “aTag” evaluates to a capital letter. This is a SmartSet “command”.
- If the host sends a message to ask the controller how it is configured, then “aTag” evaluates to a lower case letter. This is a SmartSet “query”.
- If the controller sends a message to answer the host’s query, then “aTag” evaluates to a capital letter. This is a SmartSet “response”.

Not all messages are available as both commands and responses, but those that are use the same syntax for the two forms. This provides a strong method to enforce forward and backward compatibility. The host should query the current settings of parameters, change specific values in the response and send the modified message back to the controller as a command. It doesn't matter if the host program recognizes all the features the controller supports or if the controller supports all features that the host program recognizes. They can still exchange data about the features they support in common.

List of Messages

Acknowledgement: ‘a’, ‘A’

Acknowledgement query			
Offset	Length	Name	Usage
0	1	aTag	Lower case ‘a’
1	7		Unused field. Fill with nulls.

Using an acknowledgement query, the host asks whether the controller has warnings to report. The controller sends four error codes in its response.

Acknowledgement command			
No command is available.			

If the host attempts to send an acknowledgement command, using the tag ‘A’, the controller sends an acknowledgement response asserting a “no command available” warning.

Acknowledgement response			
Offset	Length	Name	Usage
0	1	aTag	Capital ‘A’
1	1	bError0	Four error codes are reported. Values are described in table below.
2	1	bError1	
3	1	bError2	
4	1	bError3	
5	3		Unused field. Fill with nulls.

After it sends an acknowledgement response, the controller clears its internal record of warnings. If more than four errors occur between acknowledgement responses, the controller is permitted to drop the average.

The controller sends an acknowledgment response when it finishes processing any command.

Table 3 Error codes

ASCII code	Hexadecimal value	Decimal value	Meaning
‘0’	0x30	48	No error
‘1’	0x31	49	Divide by zero
‘2’	0x32	50	Bad input packet
‘3’	0x33	51	Bad input checksum

ASCII code	Hexadecimal value	Decimal value	Meaning
'4'	0x34	52	Input packet overrun
'5'	0x35	53	Illegal command
'6'	0x36	54	Calibration command canceled
'7'	0x37	55	Reserved
'8'	0x38	56	Bad serial setup combination
'9'	0x39	57	Configuration store invalid – initializing
':'	0x3a	58	Reserved
','	0x3b	59	Reserved
'<'	0x3c	60	Reserved
'='	0x3d	61	Reserved
'>'	0x3e	62	Reserved
'?'	0x3f	63	Reserved
'@'	0x40	64	Reserved
'A'	0x41	65	No command available
'B'	0x42	66	Unsupported in this firmware version
'C'	0x43	67	Illegal subcommand
'D'	0x44	68	Operand out of range
'E'	0x45	69	Invalid type
'F'	0x46	70	Fatal error condition
'G'	0x47	71	No query available
'H'	0x48	72	Reserved
'I'	0x49	73	Configuration store failed
'J'	0x4a	74	Reserved
'K'	0x4b	75	Self test failure
'L'	0x4c	76	Internal operation failed
'M'	0x4d	77	Measurement warning
'N'	0x4e	78	Measurement error

Report timing: 'B', 'b'

Report timing query			
Offset	Length	Name	Usage
0	1	aTag	Value is always 'b'.
1	7		Unused field. Fill with nulls.

Report timing command or response			
Offset	Length	Name	Usage
0	1	aTag	Capital 'B'
1	1	bUntouchTime	This byte variable has a range of 0 to 255. It specifies the number of 10 ms periods allowed between loss of touch detection and reporting of untouch. The default bUntouchTime is 3. Increase bUntouchTime to reduce skipping during dragging.
2	1	bRepDelay	This byte variable has a range of 0 to 255. It specifies the number of 10 ms periods guaranteed between stream touch reports. The default bRepDelay is 0. Increase bRepDelay to reduce I/O bandwidth requirements without degrading responsiveness to initial touch or untouch.
3	5		Unused field. Fill with nulls.

Calibration: 'C', 'c'

Calibration query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'c'
1	1	aSubcommand	Value is one of the following. Capital 'S' requests the value for the bmExchangeAxis setting. Lower case 'x' requests the values of offset, numerator and denominator used to calibrate the X-axis. Lower case 'y' requests the values of offset, numerator and denominator used to calibrate the Y-axis. Lower case 'z' requests the values of offset, numerator and denominator used to calibrate the Z-axis.
2	6		Unused field. Fill with nulls.

If a calibration query has aSubcommand set to a value other than those listed in the table, then the controller sends an acknowledgment response containing an "illegal subcommand" warning.

Calibration command or response – axis setting by parameter			
Offset	Length	Name	Usage
0	1	aTag	Capital 'C'
1	1	aSubcommand	Value is one of the following. Lower case 'x' sets the offset, numerator and denominator used to scale X-axis. Lower case 'y' sets the offset, numerator and denominator used to scale Y-axis. Lower case 'z' sets the offset, numerator and denominator used to scale Z-axis.
2	2	iOffset	This signed variable has a range of -32768 to 32767. Subtract iOffset from the measured coordinate to shift the calibrated coordinate along its axis. The default value is 0.
4	2	iNumerator	This signed variable has a range of -32768 to 32767. Multiply iNumerator by the difference of the measured coordinate and iOffset. The default value is 4095 for X and Y, 255 for Z.
6	2	iDenominator	This signed variable has a range of -32768 to 32767. Divide the product of iNumerator and the difference of the measured coordinate and iOffset by iDenominator. The default value is 4095 for X and Y, 255 for Z.

Calibration command or response – exchange X and Y			
Offset	Length	Name	Usage
0	1	aTag	Capital ‘C’
1	1	aSubcommand	Capital ‘S’ indicates the next argument is the exchange axis bit map.
2	1	bmExchangeAxis	Bit 0 = 1 to exchange X and Y in coordinate reports. The default value is 0. Bits 1 to 7 are reserved.
3	5		Unused field. Fill with nulls.

The controller immediately exchanges X and Y coordinates when it receives an exchange axis subcommand. The SmartSet “mode” command does not control axis exchange. (Not all controllers work this way, but they should. This makes it much easier to correct for a screen that is rotated by a quarter turn.)

Being true to its roots, a SmartSet controller should handle this command as much like the 2210 as possible. The 2210 actually exchanged X and Y in the measurement routine prior to any calibration or scaling operations. To be backward compatible, a SmartSet controller should handle the “swap” feature according to these rules.

- Exchange X and Y prior to any other calibration operation.
- To make the feature operation transparent to a user, exchange X and Y axis calibration parameters when swapping is turned on or off. When a “CS...” command is received, compare the commands “bmExchangeAxis” value with the current setting for the feature. If they are the same, do nothing. If they are different, then exchange X and Y offsets, numerators and denominators.

Calibration command – axis setting by range			
Offset	Length	Name	Usage
0	1	aTag	Capital ‘C’.
1	1	aSubcommand	Value is one of the following. Capital ‘X’ sets the minimum and maximum values corresponding to 0 and 4095 for the X-axis. Capital ‘Y’ sets the minimum and maximum values corresponding to 0 and 4095 for the Y-axis. Capital ‘Z’ sets the minimum and maximum values corresponding to 0 and 255 for the Z-axis.
2	2	iMinimum	This signed variable has a range of -32768 to 32767. It specifies the calibrated value that corresponds to a measured coordinate of 0. If iMinimum is greater than iMaximum, then the axis is inverted.

Calibration command – axis setting by range			
Offset	Length	Name	Usage
4	2	iMaximum	This signed variable has a range of -32768 to 32767. It specifies the calibrated value that corresponds to a measured coordinate of 4095.
6	2		Unused field. Fill with nulls.

Calibration command – run interactive calibration sequence			
Offset	Length	Name	Usage
0	1	aTag	Capital ‘C’
1	1	aSubcommand	Number ‘2’ starts running a two-point interactive calibration sequence.
2	6		Unused field. Fill with nulls.

No query corresponds to subcommands that set calibration by range or run interactive calibration.

If a calibration command has aSubcommand set to a value other than ‘S’, ‘X’, ‘Y’, ‘Z’, ‘x’, ‘y’, ‘z’ or ‘2’, then the controller sends an acknowledgement asserting an “illegal subcommand” error. Calibration values and settings are unchanged.

The controller does not apply calibration immediately when calibration constants are set by command or by running an interactive calibration. The SmartSet “mode” command turns the calculation on or off.

Setting by range versus setting by parameter

The host can set calibration constants using either range (iMinimum, iMaximum) or parameters (iOffset, iNumerator, iDenominator). The controller stores only the parameter values.

If it receives a set by range command, the controller converts the range into parameters. In this case, it sets the Numerator to the default value of 4095 for X or Y, 255 for Z.

The host cannot query axis range settings. These values can be calculated from the parameters.

$$iMinimum = iOffset$$

$$iMaximum = iMinimum + iDenominator$$

Z-Axis Calibration

SmartSet allows you to calibrate the Z-axis.

If the minimum Z value calibrates to a value other than zero, then the controller reports something other than zero on untouch.

Some existing controllers follow these rules better than others. Beware if you are writing an application program.

Diagnostics: 'D','d'

Diagnostics query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'd'
1	7		Unused field. Fill with nulls.

If the controller receives a diagnostic query, then it sends a diagnostic response with results from the last time a self-test was run.

The concluding acknowledgement response contains a “self test failure” warning if any diagnostic bitmap is nonzero.

Diagnostics command or response			
Offset	Length	Name	Usage
0	1	aTag	Capital 'D'
1	1	bmCom	<p>In a command, a set bit tells the controller to run the indicated diagnostic. In a response, a set bit means the test failed the last time it ran.</p> <p>Bit 0 = fId, ID test.</p> <p>Bit 1 = fCpu, CPU test.</p> <p>Bit 2 = fChecksum, ROM test.</p> <p>Bit 3 = fExtRam, External RAM test.</p> <p>Bit 4 = fConfig, Configuration store test.</p> <p>Bit 5 = fTouchscreen, Touchscreen and drive test (failure).</p> <p>Bit 6 = Reserved.</p> <p>Bit 7 = fTestIncomplete, Touchscreen and drive test (incomplete).</p>

Diagnostics command or response			
Offset	Length	Name	Usage
2	1	bmTouch	<p>In a command, a set bit tells the controller to run the indicated diagnostic. In a response, a set bit means the test failed the last time it ran.</p> <p>Bit 0 to bit 2 = ffScreenErrorCode</p> <p>0 = no error</p> <p>1 = X-axis failed</p> <p>2 = Y-axis failed</p> <p>3 = X receive (sensing) failed</p> <p>4 = Y receive (sensing) failed</p> <p>5 = X transmit (drive) failed</p> <p>6 = Y transmit (drive) failed</p> <p>7 = failure was not isolated to an axis or component</p> <p>Bit 3 = fADC, touch interface hardware test</p> <p>Bit 4 = fChecksum, ROM test</p> <p>Bit 5 = fExtRAM, External RAM test</p> <p>Bit 6 = fIntRAM, Internal RAM test</p> <p>Bit 7 = fCPU, CPU test</p>
3	1	bmExternal	<p>In a command, a set bit tells the controller to run the indicated diagnostic. In a response, a set bit means the test failed the last time it ran.</p> <p>Bit 0 = Monitor status.</p> <p>Bit 1 to Bit 7 = reserved.</p>
4	4		Unused field. Fill with nulls.

If the controller receives a diagnostic command, then it runs the tests associated with flag bits that are set to '1'. The controller then sends a diagnostic response followed by the concluding acknowledgement response. (This is a special case – most SmartSet commands don't respond with anything other than the acknowledgement.)

Some tests appear in both bmCom and bmTouch. This is a legacy from an old controller that had separate communication and touch microprocessors, each running independent tests.

TBD We need a policy for reading and reporting the bits.

- **Alternative 1: regardless of which bit appears in the command, all bits associated with a failing test are set in the diagnostic response. So, if a command sets fChecksum in bmCom and the ROM test fails, the response has bits fChecksum in bmCom and fChecksum in bmTouch both set. This could be a little weird looking if you intend to run only one test, but it's probably the cleanest solution.**

- Alternative 2: This is the same as above when reading bits from a command, but the response masks the result bitmap with the original bitmap from the command. So, if a command sets fChecksum in only bmCom, the response sets fChecksum in only bmCom. Here are two issues with this solution:
 - The result of this test is stored in case a diagnostic query arrives later. Should the stored value be the stripped down value that appeared in the response, or should it be the full mapping with all bits set?
 - What about the touchscreen test, which can send a bit field containing a failure code? Masking the failure code off could make a failing test appear to pass.
- Other alternatives?

TBD Because we didn't have a policy, various controller models behave differently. Their specs have weird accretions that should be gotten rid of. Here are some things I'd flush with extreme prejudice:

- bmExternal (MaskE) bits reflect situations external to the controller, and do not have to be 'set' to be interrogated – these bits will be valid in every response, independently of their state in a previous 'Set' operation. ***Make it act the same as other bit maps.***
- ... a bit set to 1 in the test mask byte would run the corresponding test while a bit reset to 0 would ignore the test. Exceptions to this rule are bits MaskC.7, MaskC.5, MaskT.1, and MaskT.0. None of these bits has any effect in a set command. ***Why were these bits singled out for special mention? It's really easy to map these bits into the right test.***

Emulate: 'E','e'

Emulate query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'e'
1	7		Unused field. Fill with nulls.

Emulate command or response			
Offset	Length	Name	Usage
0	1	aTag	Capital 'E'
1	1	aEnableStatus	The ASCII character aEnableStatus is '1' = The format may send special characters or additional touch reports to indicate initial touch and untouch status. '0' = Inhibit status reporting. Default aEnableStatus setting is '1' when combined with the default aFormat setting. See below.
2	1	aFormat	The ASCII character aFormat sets the touch response format as follows. '0' = Elo 140 binary '1' = Elo 140 ASCII '2' = Elo 280 binary '3' = Elo 280 ASCII '4' = Elo SmartSet binary '5' = Elo SmartSet ASCII '6' = Elo E281A-4002 binary '7' = Elo E281A-4002 ASCII '8' = Elo E271-60 ASCII 'A' = MT binary 'B' = MT binary stream 'C' = MT decimal ASCII 'D' = MT hexadecimal ASCII 'E' = MT "calibrate raw" 'F' = MT tablet Default aFormat setting is '4', Elo SmartSet binary.
3	5		Unused field. Fill with nulls.

Early model legacy controllers distinguish between “output-only” emulation and “two-way” emulation.

- In those models, output-only emulation was enabled through the SmartSet “E...” command. These controllers would continue to receive and process SmartSet commands, but they would send touches in the selected emulation format.
- “Two-way” emulation could only be enabled through a jumper setting. When emulating in this way, the controllers would no longer respond to SmartSet commands. They would respond to commands of the selected emulation format, and they would send touches in the selected format.

Because jumpers are deprecated in new controller models, implementation of the “E...” command changes. When an “E...” command selects a new format, the controller enters “two-way” emulation for that format. Each format must support these functions.

- The format must provide capability to convert back to Elo SmartSet format. If such a command is not native to the format, add one.
- The format must provide capability to store its configuration to NVRAM. If such a command is not native to the format, add one.

Filter Command – ‘F’, ‘f’

Filter query			
Offset	Length	Name	Usage
0	1	aTag	Lower case ‘f’
1	1	bTechnology	<p>The byte variable bTechnology indicates what kind of response is expected. The following values are valid.</p> <p>0x00 = The query requests the controller to fill in its technology type from the following list.</p> <p>0x30 = AccuTouch five-wire resistive</p> <p>0x31 = four-wire resistive</p> <p>0x32 = IntelliTouch surface acoustic wave</p> <p>0x33 = Carroll Touch infra-red</p> <p>0x34 = Surface capacitive</p> <p>0x35 = Projected capacitive</p> <p>0x36 = reserved</p> <p>0x37 = “Thunder” technology</p> <p>0x42 = IntelliTouch extended page 1</p> <p>0x52 = IntelliTouch extended page 2</p>
2	6		Unused field. Fill with nulls.

Filter command or response (IntelliTouch base page)			
Offset	Length	Name	Usage
0	1	aTag	Capital ‘F’
1	1	bTechnology	0x32 = IntelliTouch surface acoustic wave
2	2	wContaminantTimeout	<p>wContaminantTimeout is a word value that controls the length of time a repeating coordinate value is permitted to be measured by the controller. If the time that a coordinate value repeats in X, Y, and Z exceeds this limit, then the controller resets its reference waveform. This parameter determines the controller's tolerance of contaminants on the touchscreen. Range of this argument is {1,...,65535}, and it is expressed in time units of 10 milliseconds. The default value is 12000 (2EE0h), which is 120 seconds.</p>
4	1	bThreshold	<p>bThreshold is a byte value that programs the amount of SAW energy absorption recognized as a touch. A small value here increases touch sensitivity. A large value increases noise rejection. Range of this argument is {0,...,255}. The default value is 1.</p>

Filter command or response (IntelliTouch base page)			
Offset	Length	Name	Usage
5	1	bMinimumWidth	bMinimumWidth programs the minimum width for a touch. A small value increases sensitivity; and a large value increases noise rejection. Range of this argument is {0,..255}. The default value is 2.
6	1	bMaximumWidth	bMaximumWidth programs the maximum width for a touch. This parameter controls the rejection of multiple touches and splattered contaminants. Range of this argument is {0,..255}. The default value is 29 for 270x controllers, 22 for older models.
7	1	bFilterType	bFilterType selects the type of smoothing filter applied to the X, Y coordinate data in stream mode. The default value is 4. 00H = none 01H = moving average of two 02H = moving average of four 03H = moving average of eight 04H = moving average of sixteen

Filter command or response (IntelliTouch first extended page)			
Offset	Length	Name	Usage
0	1	aTag	Capital 'F'
1	1	bTechnology	0x42 = IntelliTouch extended page 1
2	2	wUntouchDistance	wUntouchDistance is a word value that controls distance-based untouch insertion. When the difference between successive X coordinates or Y coordinates exceeds the value of this setting, the controller inserts an untouch report into the coordinate stream. The parameter is reported in raw coordinate units with a range of 0000H to 0fffH (0 to 4095 decimal). The default value is 1000H (4096 decimal), which turns the feature off.
4	2	wFilterDistance	wFilterDistance limits the coordinate reporting lag that can accumulate in the coordinate smoothing filter. When the difference between successive X coordinates or Y coordinates exceeds this setting, the controller updates the filter's internal storage to the newest measured value. The default value is 200H (512 decimal).

Filter command or response (IntelliTouch first extended page)			
Offset	Length	Name	Usage
6	1	bmFBits0	<p>bmFBits0 is a bit-mapped byte that controls optional touch measurement features. The default value is 00H.</p> <p>Bit 0 = fDisableRelearn, turns off relearning when multiple touches are detected. This flag is supported only in model 2500G.</p> <p>Bit 1 = fEnableVectorAveraging, adds a layer of averaging to the internal touch waveform processing. This flag is supported only in models 270x, where this averaging is usually disabled.</p> <p>Bit 2 = fAlwaysUseEarlyYSweep, acquires the Y return signal prior to processing X and Y signals regardless of touchscreen size. This flag is supported only in models 270x, where early Y sweep is performed on 17-inch and larger screens.</p> <p>Bit 3 = fNeverUseEarlyYSweep, permits processing of the X signal prior to acquiring the Y return signal regardless of touchscreen size. This flag is supported only in models 270x, where early Y sweep is performed on 17-inch and larger screens.</p> <p>Bit 4 = fDisableWaveformAveraging, removes a layer of averaging from the internal touch waveform processing. This flag is supported only in models 270x, where this averaging is usually enabled.</p> <p>Bit 5 = Reserved.</p> <p>Bit 6 = Reserved.</p> <p>Bit 7 = Reserved.</p>
7	1		unused field

Filter command or response (IntelliTouch second extended page)			
Offset	Length	Name	Usage
0	1	aTag	Capital 'F'
1	1	bTechnology	0x52 = IntelliTouch extended page 2

Filter command or response (IntelliTouch second extended page)			
Offset	Length	Name	Usage
2	1	bMultitouchMode	<p>The byte variable bMultitouchMode sets the mode of multiple touch operation. This setting affects the information that is multiplexed into the high byte of the Z value in a multiple touch report. The default value is 00. This feature is implemented only in controller model 2500G.</p> <p>00H = no multiple touch</p> <p>01H = tag with touch priority. Oldest touch has highest priority.</p> <p>02H = tag with X partition. Indicate whether touch is assigned to left or right half of touchscreen. This mode is not implemented in any controller model.</p> <p>03H = tag with Y partition. Indicate whether touch is assigned to top or bottom half of touchscreen. This mode is not implemented in any controller model.</p>
3	1	bMaximumPriority	<p>bMaximumPriority is a byte variable indicating the maximum priority tag value for multiple touches. For example, set bMaximumPriority to 02H to permit up to three touches and reject when there are four touches. This feature is implemented only in controller model 2500G.</p>
4	2	wXPartition	<p>wXPartition sets the boundary between the left and right halves of the touchscreen reported in multiple touch mode 2. The default value is 800H (2048 decimal). This feature is implemented only in controller model 2500G.</p>
6	2	wYPartition	<p>wYPartition sets the boundary between the top and bottom halves of the touchscreen reported in multiple touch mode 3. The default value is 800H (2048 decimal). This feature is implemented only in controller model 2500G.</p>

Configuration - 'g'

Requests a complete dump of the controller's configuration for saving and restoring controller settings when switching between applications.

Configuration query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'g'
1	7		Unused field. Fill with nulls.

Storage requirements may be queried with the *ID* command. The number of packets in the transfer is returned in the P byte.

The packets may be sent back to the controller as individual commands to restore all controller parameters.

User Timer - 'H','h'

Timer query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'h'
1	7		Unused field. Fill with nulls.

This command is not supported by this controller.

ID Command - 'i'

Identity query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'i'
1	7		Unused field. Fill with nulls.

Identity response			
Offset	Length	Name	Usage
0	1	aTag	Capital 'I'
1	1	bTechnology	<p>The byte variable bTechnology indicates what kind of touchscreen is connected. The following values are valid.</p> <ul style="list-style-type: none"> 0x30 = AccuTouch five-wire resistive 0x31 = four-wire resistive 0x32 = IntelliTouch surface acoustic wave 0x33 = Carroll Touch infra-red 0x34 = Surface capacitive 0x35 = Projected capacitive 0x36 = reserved 0x37 = "Thunder" technology 0x42 = IntelliTouch extended page 1 0x52 = IntelliTouch extended page 2
2	1	bInterface	<p>The byte variable bInterface indicates how the controller communicates with the host computer. The following values are valid.</p> <ul style="list-style-type: none"> 0x30 = RS-232 serial 0x31 = ISA bus 0x32 = Micro Channel bus 0x33 = Apple Desktop Bus 0x34 = USB 0x35 = I2C or SMBUS

Identity response			
Offset	Length	Name	Usage
3	1	bmFeatures	<p>The bitmapped byte bmFeatures lists optional features of the controller.</p> <ul style="list-style-type: none"> Bit 0 = reserved Bit 1 = reserved Bit 2 = reserved Bit 3 = reserved Bit 4 = optional external A/D converter Bit 5 = optional extended RAM Bit 6 = optional RAM available Bit 7 = Z axis
4	1	bmMinorRevision	<p>The bit mapped byte variable bmMinorRevision contains two four-bit values relating to firmware revision level.</p> <ul style="list-style-type: none"> Bit 0 to bit 3 = ffProgramMinorRevision Bit 4 to bit 7 = ffBranchIdentifier
5	1	bmMajorRevision	<p>The bit mapped byte variable bmMajorRevision contains two four-bit values relating to firmware revision level.</p> <ul style="list-style-type: none"> Bit 0 to bit 3 = ffProgramMajorRevision Bit 4 to bit 7 = ffBranchRevision
6	1	bPacketCount	<p>The byte variable bPacketCount is the number of responses that are sent to a SmartSet configuration query ('g').</p>

Identity response			
Offset	Length	Name	Usage
7	1	bModelClass	<p>The byte variable bModelClass identifies the controller model. The following values are valid.</p> <ul style="list-style-type: none"> 0x01 = AccuTouch model 2210 0x03 = IntelliTouch model 2310 0x05 = IntelliTouch model 2310B 0x06 = IntelliTouch model 2500S 0x07 = IntelliTouch model 2500U 0x08 = AccuTouch model 3000U 0x09 = Carroll touch IR model 4000U 0x0a = Carroll touch IR model 4000S 0x0d = Carroll touch IR SmartSet Lite serial 0x0e = AccuTouch serial chipset 0x0f = IntelliTouch model 2500G 0x20 = IntelliTouch model 2701 or model 2700 0x30 = Carroll touch IR model 4500U 0x31 = Carroll touch IR model 4500S 0x32 = Carroll touch IR model 4501 0x41 = model 5000 and model 5010 surface capacitive 0x50 = model 2216 AccuTouch or 4-wire resistive 0xff = generic Elo controller model

Jumpers Command - 'j'

Jumpers query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'j'
1	7		Unused field. Fill with nulls.

Jumpers response			
Offset	Length	Name	Usage
0	1	aTag	Capital 'J'
1	1	bTechnology	<p>The byte variable bTechnology indicates what kind of touchscreen is connected. The following values are valid.</p> <ul style="list-style-type: none"> 0x30 = AccuTouch five-wire resistive 0x31 = four-wire resistive 0x32 = IntelliTouch surface acoustic wave 0x33 = Carroll Touch infra-red 0x34 = Surface capacitive 0x35 = Projected capacitive 0x36 = reserved 0x37 = "Thunder" technology 0x42 = IntelliTouch extended page 1 0x52 = IntelliTouch extended page 2
2	1	bInterface	<p>The byte variable bInterface indicates how the controller communicates with the host computer. The following values are valid.</p> <ul style="list-style-type: none"> 0x30 = RS-232 serial 0x31 = ISA bus 0x32 = Micro Channel bus 0x33 = Apple Desktop Bus 0x34 = USB 0x35 = I2C or SMBUS

Jumpers response			
Offset	Length	Name	Usage
3	1	bmJumper	<p>The bitmapped byte bmJumper shows which hardware jumpers are installed. Jumper usage and bit mapping are not consistent between various controller models.</p> <p>Bit 0 = reserved</p> <p>Bit 1 = Jumper J1 is installed, if the jumper exists</p> <p>Bit 2 = Jumper J2 is installed, if the jumper exists</p> <p>Bit 3 = Jumper J3 is installed, if the jumper exists</p> <p>Bit 4 = Jumper J4 is installed, if the jumper exists</p> <p>Bit 5 = Jumper J5 is installed, if the jumper exists</p> <p>Bit 6 = Jumper J6 is installed, if the jumper exists</p> <p>Bit 7 = Jumper J7 is installed, if the jumper exists</p>
4	1	bmCrossJumper	<p>The bitmapped byte bmCrossJumper shows which hardware jumpers are cross connected. Jumper usage and bit mapping are not consistent between various controller models.</p> <p>Bit 0 = reserved</p> <p>Bit 1 = Jumper J1 connects to J2, if the jumpers exist</p> <p>Bit 2 = Jumper J2 connects to J3, if the jumpers exist</p> <p>Bit 3 = Jumper J3 connects to J4, if the jumpers exist</p> <p>Bit 4 = Jumper J4 connects to J5, if the jumpers exist</p> <p>Bit 5 = Jumper J5 connects to J6, if the jumpers exist</p> <p>Bit 6 = Jumper J6 connects to J7, if the jumpers exist</p> <p>Bit 7 = reserved</p>
5	1	bBaudRate	indicates the baud rate if the interface is RS232 serial
6	1	aSerialHandshake	<p>'0' = Handshaking is disabled due to a jumper setting</p> <p>'1' = Handshaking is enabled</p>
7	1	aPreferredFormat	'1' = Preferred format is SmartSet binary

Key Command - 'K','k'

Key query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'k'
1	7		Unused field. Fill with nulls.

Low Power - 'L','l'

Low-power query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'l'
1	7		Unused field. Fill with nulls.

Mode Command - 'M','m'

Mode query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'm'
1	7		Unused field. Fill with nulls.

Mode command			
Offset	Length	Name	Usage
0	1	aTag	Capital 'M'
1	1	bBinaryAscii	Set to 0x00 for binary.
2	1	bmModeFlags1	<p>The bit mapped byte bmModeFlags1 indicates which touch reporting features to enable. Bits have these meanings.</p> <ul style="list-style-type: none"> Bit 0 = Report the initial location of a user touch. Bit 1 = Report the stream of locations during a continuing touch. Bit 2 = Report the last measured touch after a user stops touching. Bit 3 = Enable reporting of multiple simultaneous touches. Bit 4 = reserved Bit 5 = reserved Bit 6 = Check the range of coordinate values. If X or Y is outside its native range due to calibration, then flag the condition. Bit 7 = Enable Z axis measurement if the controller supports Z measurement.

Mode command			
Offset	Length	Name	Usage
3	1	bmModeFlags2	<p>The bit mapped byte bmModeFlags2 indicates which touch reporting features to enable. Bits have these meanings.</p> <p>Bit 0 = reserved</p> <p>Bit 1 = Trim the range of coordinates reported. If a coordinate value is outside the native range, then report the corresponding maximum or minimum value. Enabling “trim” mode automatically enables the range checking feature controlled by bit 6 of bmModeFlags1.</p> <p>Bit 2 = Calibrate reported coordinate values using calibration values set by the ‘C’ command.</p> <p>Bit 3 = Scale the range of reported coordinate values using scaling values set by the ‘S’ command.</p> <p>Bit 4 = reserved (Zoned reporting)</p> <p>Bit 5 = reserved (Validated reporting)</p> <p>Bit 6 = Track the value of previous reported coordinates. Send a new report only if it differs from the last reported value in X, Y, Z or status.</p> <p>Bit 7 = reserved (Differential reporting)</p>
4	4	unused field	Set to nulls

Mode command (optional ASCII format)			
Offset	Length	Name	Usage
0	1	aTag	Capital ‘M’
1	1	bBinaryAscii	Set to 0x30 for ASCII. Most controllers do not support ASCII formatted commands.

Mode command (optional ASCII format)			
Offset	Length	Name	Usage
2	6	aModeString	<p>The string may contain up to six ASCII characters to enable the following features.</p> <p>‘B’ = Enable (bounds) range checking.</p> <p>‘C’ = Enable calibration.</p> <p>‘D’ = reserved (Differential reporting)</p> <p>‘I’ = Report the user’s initial touch location</p> <p>‘M’ = Enable scaling (map)</p> <p>‘P’ = Enable coordinate range trimming. This setting automatically enables range checking</p> <p>‘S’ = Report stream touches</p> <p>‘T’ = Enable previous coordinate tracking.</p> <p>‘U’ = Report untouch</p> <p>‘Z’ = Measure Z axis values</p> <p>Any other character value terminates the mode setting sting.</p>

Mode response			
Offset	Length	Name	Usage
0	1	aTag	Capital ‘M’
1	1	bBinaryAscii	Set to 0x00 for binary.
2	1	bmModeFlags1	<p>The bit mapped byte bmModeFlags1 indicates which touch reporting features are enabled. The bitmap corresponds to the status field in a SmartSet touch report. In a mode response, the bits have these meanings.</p> <p>Bit 0 = Initial location of a user touch is reported.</p> <p>Bit 1 = The stream of locations during a continuing touch is reported.</p> <p>Bit 2 = The last measured touch location is reported after a user stops touching.</p> <p>Bit 3 = Multiple simultaneous touches are reported.</p> <p>Bit 4 = reserved</p> <p>Bit 5 = reserved</p> <p>Bit 6 = Range checking is enabled. If X or Y falls outside its native range due to calibration, then the condition is flagged.</p> <p>Bit 7 = Z axis measurement is enabled.</p>

Mode response			
Offset	Length	Name	Usage
3	1	bmModeFlags2	<p>The bit mapped byte bmModeFlags2 indicates which touch reporting features are enabled. Bits have these meanings.</p> <p>Bit 0 = reserved</p> <p>Bit 1 = Out of range coordinates are “trimmed”. If a coordinate value is outside the native range, then report the corresponding maximum or minimum value. Enabling “trim” mode automatically enables the range checking feature controlled by bit 6 of bmModeFlags1.</p> <p>Bit 2 = Reported coordinate values are calibrated using values set by the ‘C’ command.</p> <p>Bit 3 = Reported coordinate values are scaled using values set by the ‘S’ command.</p> <p>Bit 4 = reserved (Zoned reporting)</p> <p>Bit 5 = reserved (Validated reporting)</p> <p>Bit 6 = Previous reported coordinates are “tracked”. A report is sent only if it differs from the last reported value in X, Y, Z or status.</p> <p>Bit 7 = reserved (Differential reporting)</p>
4	4	unused field	Set to nulls

Nonvolatile RAM Command - 'N'

Saves or retrieve controller settings in the on-board nonvolatile memory (NVRAM). NVRAM is used to store power-on defaults.

Nonvolatile command			
Offset	Length	Name	Usage
0	1	aTag	Capital 'N'
1	1	bWriteEnable	Set to 0x01 to write the current run time values into nonvolatile storage. Set to 0x00 to read from nonvolatile storage into the run time values.
2	1	bmCategories	This bit mapped byte selects which categories of data are to be written to or read from nonvolatile storage. Data from all categories can be stored in response to a single command. Bits have the following meanings. Bit 0 = communication, touch mode and other set up information. Bit 1 = calibration values Bit 2 = scaling values Bit 3 to Bit 7 = reserved
3	1	bPage	This optional byte value selects from multiple pages in nonvolatile storage for certain categories of data. Calibration and scaling values are stored in two pages indicated as 0x00 and 0x01. At power on, calibration and scaling values are retrieved from page 0x00 only. There is only one set up storage page.
4	4	unused field	set to nulls

Owner Command - 'o'

Owner query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'o'
1	7		Unused field. Fill with nulls.

Owner response			
Offset	Length	Name	Usage
0	1	aTag	Capital 'O'
1	7	aOwnerString	Fixed value = "EloInc."

Parameter Command - 'P','p'

This command changes the baud rate of the controller.

Parameter query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'p'
1	7		Unused field. Fill with nulls.

Parameter command or response (for RS-232 serial interface)			
Offset	Length	Name	Usage
0	1	aTag	Capital 'P'
1	1	bInterface	<p>The byte variable bInterface indicates how the controller communicates with the host computer. The following values are valid.</p> <ul style="list-style-type: none"> 0x30 = RS-232 serial 0x31 = ISA bus 0x32 = Micro Channel bus 0x33 = Apple Desktop Bus 0x34 = USB 0x35 = I2C or SMBUS

Parameter command or response (for RS-232 serial interface)			
Offset	Length	Name	Usage
2	1	bmCharacterFrame	<p>The bitmapped byte bmCharacterFrame controls the serial character format.</p> <ul style="list-style-type: none"> Bit 2, 1 and 0 = Baud rate bit field <ul style="list-style-type: none"> 0 = 300 baud 1 = 600 baud 2 = 1200 baud 3 = 2400 baud 4 = 4800 baud 5 = 9600 baud 6 = 19200 baud 7 = (optional) faster baud if supported Bit 3 = Set to send 7-bit characters Bit 4 = Set for two stop bits Bit 5 = Set to enable parity Bit 7 and 6 = Parity type bit field <ul style="list-style-type: none"> 0 = even parity 1 = odd parity 2 = always mark parity 3 = always space parity
3	1	bmHandshake	<p>The bitmapped byte bmHandshake controls the RS-232 handshaking configuration.</p> <ul style="list-style-type: none"> Bit 0 = Require host to send correct check sum in SmartSet commands. Bit 1 = Respond to XON and XOFF character handshaking. Bit 2 = Respond to RTS and CTS hardware handshaking. Bit 3 = Invert sense of RTS control. Bit 4 = reserved Bit 5 = reserved Bit 6 = reserved Bit 7 = Enable echo (half duplex).
4	4		Unused field. Fill with nulls.

Quiet Command - 'Q','q'

Quiet query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'q'
1	7		Unused field. Fill with nulls.

Quiet command or response			
Offset	Length	Name	Usage
0	1	aTag	Capital 'Q'
1	1	bmQuietCategories	<p>The bit mapped byte variable bmQuietCategories indicates which response types are disabled.</p> <ul style="list-style-type: none"> Bit 0 = All response types including responses to SmartSet commands. Bit 1 = Timer expiration responses Bit 2 = Touch responses Bit 3 to 7 = reserved
2	6		Unused field. Fill with nulls.

If bit 0 of bmQuietCategories is set, then the controller does not respond with an acknowledgement. After setting this option, the controller does not acknowledge or respond to any command.

Reset Command - 'R'

Rest query
No query is available.

Reset command			
Offset	Length	Name	Usage
0	1	aTag	Capital 'R'
1	1	bResetAction	<p>The byte variable bResetAction commands the controller to perform one of the following tasks.</p> <p>0x00 or '0' = "Cold boot"; this has the same effect as cycling power on the controller.</p> <p>0x01 or '1' = "Warm boot"; restart the firmware program, but do not re-initialize operational settings.</p> <p>0x02 or '2' = Reprogram the nonvolatile configuration store to default operational settings, and restart the firmware program using these settings.</p> <p>0x03 or '3' = Jump to the "flash loader" if this controller supports field reprogramming.</p>
2	6		Unused field. Fill with nulls.

If bResetAction commands a cold boot, the command does not respond with an acknowledgement.

Some controller models do not respond with an acknowledgement when commanded to reprogram their nonvolatile configuration.

If the application does not allow the host to monitor -CTS, then approximately 5 seconds should be allowed for the reset procedure to take place.

Scaling Command - 'S','s'

Scaling query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 's'
1	1	aSubcommand	Value is one of the following. Capital 'S' requests the value for the bmInvertAxis setting. Lower case 'x' requests the values of offset, numerator and denominator used to calibrate the X-axis. Lower case 'y' requests the values of offset, numerator and denominator used to calibrate the Y-axis. Lower case 'z' requests the values of offset, numerator and denominator used to calibrate the Z-axis.
2	6		Unused field. Fill with nulls.

If a scaling query has aSubcommand set to a value other than those listed in the table, then the controller sends an acknowledgment response containing an “illegal subcommand” warning.

Scaling command or response – axis setting by factor			
Offset	Length	Name	Usage
0	1	aTag	Capital 'S'
1	1	aSubcommand	Value is one of the following. Lower case 'x' sets the offset, numerator and denominator used to scale X-axis. Lower case 'y' sets the offset, numerator and denominator used to scale Y-axis. Lower case 'z' sets the offset, numerator and denominator used to scale Z-axis.
2	2	iOffset	This signed variable has a range of -32768 to 32767. Add iOffset to the coordinate after multiplying by iNumerator and dividing by iDenominator. The default value is 0.
4	2	iNumerator	This signed variable has a range of -32768 to 32767. Multiply iNumerator by coordinate. The default value is 4095 for X and Y, 255 for Z.
6	2	iDenominator	This signed variable has a range of -32768 to 32767. Divide the product of iNumerator and the by iDenominator. The default value is 4095 for X and Y, 255 for Z.

Scaling command or response – invert X, Y and Z			
Offset	Length	Name	Usage
0	1	aTag	Capital ‘S’
1	1	aSubcommand	Capital ‘S’ indicates the next argument is the exchange axis bit map.
2	1	bmInvertAxis	Bit 0 = 1 to invert X values in coordinate reports. The default value is 0. Bit 1 = 1 to invert Y values in coordinate reports. The default value is 0. Bit 2 = 1 to invert Z values in coordinate reports. The default value is 0. Bits 3 to 7 are reserved.
3	5		Unused field. Fill with nulls.

The controller inverts X, Y and Z coordinates when it receives a scaling invert axis subcommand. Inversion is an independent operation that does not affect the scaling parameters iOffset, iNumerator, and iDenominator.

In some legacy controller models, the SmartSet “mode” command controls axis inversion. These models require scaling to be enabled because axis inversion was accomplished by changing the scaling parameters. This practice is not recommended in new designs.

Scaling command – axis setting by range			
Offset	Length	Name	Usage
0	1	aTag	Capital ‘S’.
1	1	aSubcommand	Value is one of the following. Capital ‘X’ sets the minimum and maximum values corresponding to 0 and 4095 for the X-axis. Capital ‘Y’ sets the minimum and maximum values corresponding to 0 and 4095 for the Y-axis. Capital ‘Z’ sets the minimum and maximum values corresponding to 0 and 255 for the Z-axis.
2	2	iMinimum	This signed variable has a range of -32768 to 32767. It specifies the scaled value that corresponds to a measured coordinate of 0. If iMinimum is greater than iMaximum, then the axis is inverted.
4	2	iMaximum	This signed variable has a range of -32768 to 32767. It specifies the scaled value that corresponds to a measured coordinate of 4095 for X or Y, 255 for Z.
6	2		Unused field. Fill with nulls.

If a scaling command has aSubcommand set to a value other than 'S', 'x', 'y', 'z', 'X', 'Y' or 'Z', then the controller sends an acknowledgment response containing an "illegal subcommand" warning. The scaling settings are not changed.

There is no way to directly query the iMinimum and iMaximum values. These values can be calculated by these formulas.

$$iMinimum = iOffset$$

$$iMaximum = iMinimum + iNumerator$$

Scaling and Axis Inversion are disabled by factory default.

Touch Command - 't'

Touch query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 't'
1	7		Unused field. Fill with nulls.

Upload Command - 'U','u'

The SmartSet upload command provides access to debugging and troubleshooting features. Its syntax is model specific.

Upload query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'u'
1	1	bSubcommand	
2	1	bMode	
3	5		Unused field. Fill with nulls.

Upload command			
Offset	Length	Name	Usage
0	1	aTag	Capital 'U'
1	1	bSubcommand	
2	1	bArgument1	
3	5		Unused field. Fill with nulls.

Upload response			
Offset	Length	Name	Usage
0	1	aTag	Capital 'U'
1	1	bSubcommand	
2	1	bArgument1	
3	1	bArgument2	(optional) Set to null if not used.
4	1	bArgument3	(optional) Set to null if not used.
5	3		Unused field. Fill with nulls.

Subcommand 'S' - Query size of block

Upload size query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'u'
1	1	bSubcommand	Capital 'S'
2	1	bBlock	The byte variable bBlock is a zero-based index for a set of values that describe the controller's interface to the touchscreen. Block descriptions appear in a later section of this document.
3	5		Unused field. Fill with nulls.

Upload size response			
Offset	Length	Name	Usage
0	1	aTag	Capital 'U'
1	1	bSubcommand	Capital 'S'
2	1	bBlock	Echoes the variable bBlock in the query.
3	1	bBlockLengthLSB	Least significant byte of the number of SmartSet packets needed to transmit the block.
4	1	bBlockCount	The total number of blocks that may be read back from this controller. The variable bBlock has the range from zero to one less than bBlockCount.
5	1	bBlockLengthMSB	Most significant byte of the number of SmartSet packets needed to transmit the block. This is a small kludge to support block lengths greater than 255 while maintaining compatibility with early IntelliTouch SmartSet controller models.
6	2		Unused field. Fill with nulls.

This subcommand is only available as a query.

Subcommand 'C' - Query content of block.

Upload content query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'u'
1	1	bSubcommand	Capital 'C'
2	1	bBlock	The byte variable bBlock is a zero-based index for a set of values that describe the controller's interface to the touchscreen. Block descriptions appear in a later section of this document.

Upload content query			
Offset	Length	Name	Usage
3	5		Unused field. Fill with nulls.

This subcommand is only available as a query.

IntelliTouch upload block content

The response to the block content query is a series of touch packets. Length and content of blocks varies depending on the controller model. The current block definitions are listed below.

Upload block 0 content response: Waveform amplitude and timing				
aTag	bStatus	iX	iY	iZ
Capital 'T'	0x00	wXGainRange, the X receive channel high frequency gain	wYGainRange, the Y receive channel high frequency gain	0x00
Capital 'T'	0x00	wXGainLevel, the X receive channel post detection gain	wYGainLevel, the Y receive channel post detection gain	0x01
Capital 'T'	0x00	wXWaveDelay, the time from sending burst until the leading edge of the X receive pulse in units of sample periods	wYWaveDelay, the time from sending burst until the leading edge of the Y receive pulse in units of sample periods	0x02
Capital 'T'	0x00	wXWaveDuration, the time from the leading edge to the trailing edge of the X receive pulse in units of sample periods	wYWaveDuration, the time from the leading edge to the trailing edge of the Y receive pulse in units of sample periods	0x03
Capital 'T'	0x00	wXBurstLength, the number of cycles of burst sent to the X channel	wYBurstLength, the number of cycles of burst sent to the Y channel	0x04
Capital 'T'	0x00	wXZeroOffset, the A/D converter zero offset in units of A/D least significant bits	wYZeroOffset, (optional) the A/D converter zero offset in units of A/D least significant bits if X and Y can have different offsets	0x05

Upload block 1 content response: Valid and invalid measurement counters				
aTag	bStatus	iX	iY	iZ
Capital 'T'	0x00	wValidTouchCount, the number of correctly formed touches measured	wInterruptedTouchCount, the number of touches not measured due to interruption by a higher priority task	0x00

Upload block 1 content response:				
Valid and invalid measurement counters				
aTag	bStatus	iX	iY	iZ
Capital 'T'	0x00	wIncompleteTouchCount, the number of X's without Y, Y's without X or measurements with Z equal to zero	wMalformedTouchCount, the number of touches where X or Y was too wide or too narrow	0x01
Capital 'T'	0x00	wLearnCount, the number of times reference waveforms were learned	wAutoSizeCount, the number of times the screen size was calculated	0x02
Capital 'T'	0x00	wTouchEventCount, the number of touch sequences starting with initial touch, containing any number of stream touches and ending with untouch	unused field set to zero	0x03
Capital 'T'	0x00	wXSignalToNoiseRatio, an estimate of X return signal SNR	wYSignalToNoiseRatio, an estimate of Y return signal SNR	0x04

After this block response is sent, the controller clears all count values to zero.

Upload block 2 content response:				
Reference waveforms				
aTag	bStatus	iX	iY	iZ
Capital 'T'	0x00	wXReference value at sample period 0, the expected size of the return signal with no touch	wYReference value at sample period 0	0x00
The count of response packets sent is the larger of wXWaveDuration and wYWaveDuration from block 0. The shorter waveform is padded at the end with zeros. The Z value marks the sequencing of packets from 0 to the packet count less one.				
Capital 'T'	0x00	wXReference value at sample period N	wYReference value at sample period N	N

Upload block 3 content response:				
Touch threshold waveforms				
aTag	bStatus	iX	iY	iZ
Capital 'T'	0x00	wXThreshold value at sample period 0, a measurement less than this value indicates a touch	wYThreshold value at sample period 0	0x00

Upload block 3 content response:				
Touch threshold waveforms				
aTag	bStatus	iX	iY	iZ
The count of response packets sent is the larger of wXWaveDuration and wYWaveDuration from block 0. The shorter waveform is padded at the end with zeros. The Z value marks the sequencing of packets from 0 to the packet count less one.				
Capital 'T'	0x00	wXThreshold value at sample period N	wYThreshold value at sample period N	N

Upload block 4 content response:				
Frequency-related settings				
aTag	bStatus	iX	iY	iZ
Capital 'T'	0x00	wXSamplePitch in thousandths of a millimeter	wYSamplePitch in thousandths of a millimeter	0x00
Capital 'T'	0x00	dwXBurstFrequency		0x01
Capital 'T'	0x00	dwYBurstFrequency		0x02
Capital 'T'	0x00	iXBandPassTuning, positive or negative steps relative to nominal	iYBandPassTuning, positive or negative steps relative to nominal	0x03
Capital 'T'	0x00	wXLowPassFrequency, in kiloHertz	wYLowPassFrequency, in kiloHertz	0x04
Capital 'T'	0x00	wADCOffset	wDetectorOffset	0x05
Capital 'T'	0x00	wBandPassFactoryTrim	wGainFactoryTrim	0x06

Carroll Touch upload block content

The response to the block content query is one or more packets formatted as described here.

Upload block 0 content response:			
Frame information			
Offset	Length	Name	Usage
0	1	aTag	Capital 'U'
1	1	bSubcommand	Capital 'C'
2	1	bBlock	0x00, echoes the variable bBlock in the query.
3	1	bXBeamCount	The variable bXBeamCount is the number of beams measuring the X axis.
4	1	bYBeamCount	The variable bYBeamCount is the number of beams measuring the Y axis.

Upload block 0 content response:			
Frame information			
Offset	Length	Name	Usage
5	1	bXMaximumGain	The variable bXMaximumGain is the maximum gain setting for any X-axis photodetector.
6	1	bYMaximumGain	The variable bYMaximumGain is the maximum gain setting for any Y-axis photodetector.
7	1	bSequenceCount	0x00

Upload block 1 content response:			
Diagnostic information			
Offset	Length	Name	Usage
0	1	aTag	Capital 'U'
1	1	bSubcommand	Capital 'C'
2	1	bBlock	0x01, echoes the variable bBlock in the query.
3	1	bmDiagnostic1	Bit 0 = reserved Bit 1 = SFP bad command, used on EEPROM reads Bit 2 = reserved Bit 3 = failed beam Bit 4 to 7 = reserved
4	1	bmDiagnostic2	Bit 0 to 7 = reserved
5	1	bmDiagnostic3	Bit 0 = high ambient light level Bit 1 to 7 = reserved
6	1	bmDiagnostic4	Bit 0 = checksum error Bit 1 to 7 = reserved
7	1	bSequenceCount	0x00

Upload block 2 content:				
Failed beam report				
aTag	bSubcommand	bBlock	dwBeamStates	bSequenceCount
Capital 'U'	Capital 'C'	0x02	Failed beam status for beams 0 through 31. Bit = '1' indicates a failure.	0x00
Capital 'U'	Capital 'C'	0x02	Failed beam status for beams 32 through 63. Bit = '1' indicates a failure.	0x01

Upload block 2 content:				
Failed beam report				
aTag	bSubcommand	bBlock	dwBeamStates	bSequenceCount
Capital 'U'	Capital 'C'	0x02	Failed beam status for beams 64 through 95. Bit = '1' indicates a failure.	0x02
Capital 'U'	Capital 'C'	0x02	Failed beam status for beams 96 through 127. Bit = '1' indicates a failure.	0x03

Upload block 3 content:							
Beam gains response							
aTag	bSubcommand	bBlock	bGain0	bGain1	bGain2	bGain3	bSequenceCount
Capital 'U'	Capital 'C'	0x03	Beam 0 gain	Beam 1 gain	Beam 2 gain	Beam 3 gain	0x00
Capital 'U'	Capital 'C'	0x03	Beam N gain	Beam N+1 gain	Beam N+2 gain	Beam N+3 gain	0x01,...,0x1e
Each response reports the gain value for 4 beams. If gain setting for a particular beam is invalid, as it will be for a failed beam, then the most significant value of gain is set. There are 32 beam gain response packets delivering a total of 128 gain values.							
Capital 'U'	Capital 'C'	0x03	Beam 124 gain	Beam 125 gain	Beam 126 gain	Beam 127 gain	0x1f

Upload block 4 content:				
Version report				
aTag	bSubcommand	bBlock	aStringSegment	bSequenceCount
Capital 'U'	Capital 'C'	0x04	"-PPP"	0x00
Capital 'U'	Capital 'C'	0x04	"P-VV"	0x01
Capital 'U'	Capital 'C'	0x04	"-R-p"	0x02
Capital 'U'	Capital 'C'	0x04	"ppp-"	0x03
Capital 'U'	Capital 'C'	0x04	"vv-r"	0x04

Upload block 4 content:				
Version report				
aTag	bSubcommand	bBlock	aStringSegment	bSequenceCount
<p>The five responses in this block send a total of 20 characters. The 20 characters report the firmware and frame part number, version and revision level.</p> <p>Concatenating the five string segments gives “-PPPP-VV-R-pppp-vv-r”.</p> <p>PPPP = the firmware part number, 2920-PPPP</p> <p>VV = the firmware two-digit version number</p> <p>R = the firmware one-digit revision level</p> <p>pppp = the frame part number</p> <p>vv = the frame two-digit version number</p> <p>r = the frame one-digit revision level</p>				

Subcommand 'M' - Set or query mode

Upload mode query			
Offset	Length	Name	Usage
0	1	aTag	Lower case ‘u’
1	1	bSubcommand	Capital ‘M’
2	6		Unused field. Fill with nulls.

Upload mode response			
Offset	Length	Name	Usage
0	1	aTag	Capital ‘U’
1	1	bSubcommand	Capital ‘M’

Upload mode response			
Offset	Length	Name	Usage
2	1	bMode	<p>Values of bMode are from 0 up to one less than bModeCount. The mode settings are model specific. For example, the following values are those of the 2500S.</p> <p>0x00 = Normal measurement mode</p> <p>0x01 = Factory alignment mode</p> <p>0x02 = Learn reference waveforms. After learning reference waveforms, the controller will automatically shift to mode 0.</p> <p>0x03 = Auto size the touchscreen. After auto sizing, the controller will automatically shift to mode 2 and then to mode 0.</p> <p>0x04 = Test mode. In this mode, the controller periodically sends X and Y burst, and it captures signals, but it does not process signals to find touches.</p> <p>0x05 = Quiet mode. The controller microprocessor continues to run, but it does not burst the screen or acquire signals.</p> <p>0x06 = Shutdown mode. The controller microprocessor enters its low power shutdown mode.</p>
3	1	bModeCount	The total number of mode settings available.
4	4		Unused field. Fill with nulls.

Upload mode command			
Offset	Length	Name	Usage
0	1	aTag	Capital 'U'
1	1	bSubcommand	Capital 'M'
2	1	bMode	Values of bMode are those described above for the upload mode response.
3	5		Unused field. Fill with nulls.

Subcommands 'W' and 'N' - Set or query Serial Number

Upload serial number query			
Offset	Length	Name	Usage
0	1	aTag	Lower case 'u'
1	1	bSubcommand	Capital 'N'
2	1	bPage	If bPage is in the range 0x01, ..., 0x04, then the controller sends a paged serial number response. If bPage is outside that range, then the controller sends a "legacy" serial number response.

Upload serial number query			
Offset	Length	Name	Usage
3	5		Unused field. Fill with nulls.

Upload serial number response, “legacy” format			
Offset	Length	Name	Usage
0	1	aTag	Capital ‘U’
1	1	bSubcommand	Capital ‘N’
2	6	aNumberString	Six characters or bytes containing the programmed serial number.

Upload serial number response, paged format			
Offset	Length	Name	Usage
0	1	aTag	Capital ‘U’
1	1	bSubcommand	Capital ‘N’
2	1	bPage	bPage is a number in the range 0x01,...,0x04 showing the page in which the following field is stored. It echoes the value of bPage in the query.
3	5	aPageString	Five characters or bytes containing the value programmed for this page.

Upload serial number command, “legacy” format			
Offset	Length	Name	Usage
A serial number can be written only by a sequence of two commands. First is a “write unlock” command.			
0	1	aTag	Capital ‘U’
1	1	bSubcommand	Capital ‘W’
2	6		Unused field. Fill with nulls.
Second is a command setting the actual serial number value. This command is valid only if it immediately follows a write unlock command.			
0	1	aTag	Capital ‘U’
1	1	bSubcommand	Capital ‘N’
2	6	aNumberString	Six characters or bytes containing the programmed serial number. The value of the first byte must not be in the range 0x00 through 0x04.

Upload serial number command, paged format			
Offset	Length	Name	Usage
A serial number can be written only by a sequence of two commands. First is a “write unlock” command.			
0	1	aTag	Capital ‘U’
1	1	bSubcommand	Capital ‘W’
2	6		Unused field. Fill with nulls.
Second is a command setting the actual serial number value. This command is valid only when it immediately follows a write unlock command.			
0	1	aTag	Capital ‘U’
1	1	bSubcommand	Capital ‘N’
2	1	bPage	bPage is a number in the range 0x01,...,0x04 showing the page in which the following field is stored. It echoes the value of bPage in the query.
3	5	aPageString	Five characters or bytes containing the value programmed for this page.

The CTR-250000-IT-SER-0G serial number is written to NVRAM using the command sequence ‘UW’ (set ‘write enable’), ‘UNp’ (write 5 bytes into NVRAM). At any time the ‘uNp’ query may be issued to retrieve the specified five byte field previously saved.

Four independent fields of five bytes each are available for storage in the controller. The format of the data storage is not imposed by the firmware with the following exception: The first page (the page specified with the lower two bits of that field being 00) is used to construct the plug and play serial number. Thus the plug and play serial number is generated as follows:

```
(<SOH>$ELO2500\06abcde0\\Elo TouchSystems IntelliTouch 2500Gzz)
```

where <SOH> is the character ASCII 01

abcde is the five bytes from the UN0 storage area

zz is the checksum of the string between (and).

Note that plug and play is not implemented in the CTR-25000-IT-SER-SG controller.

An example:

The vendor serial number used on controllers has 13 ASCII characters, ie 13 bytes. The command format of the ‘UN0’ and ‘Un0’ commands has room for 5 bytes. The 5 bytes are the 4th character of the vendor code, followed by the last 4 characters of the vendor code. Therefore a typical vendor serial number would be the following ASCII string:

```
SSTAA6B123456
```

The first three characters (SST) are the vendor code.

The fourth character is always A for the CTR-250000-IT-SER-00

The fifth character (A) is A through L for month of manufacture January through December.

The sixth character (6) is the last digit of the year of manufacture.

The seventh character (B) is the revision.

The last 6 characters are the numeric part of the serial number.

The serial number transmitted by the SmartSet command would be the following 5 bytes:

41h, 32h, 33h, 34h, 35h, 36h This is 'A3456', from the above serial number.

Therefore the resulting PnP identity string would be:

(<SOH>\$ELO2500\06A34560\\Elo TouchSystems IntelliTouch 2500Gzz)

Why go to such trouble to have two serial numbers?

Subcommand 'U' - Debug monitor commands.

This subcommand is used to get or set values in the controller's internal or external memory. Syntax of the command is described in the following command frame.

Set:

0	1	2	3	4	5	6	7
'U'	'U'	'S' or 'G'	'I' or 'X'	Addr (LSB) (MSB)	Value	

S or G indicates that the command will either set (overwrite) the value of one byte in memory or get the values of multiple memory bytes. The response to a Get operation is a sequence of touch packets with memory byte values reported in X and Y. The Z coordinate of each packet indicates the address of the byte reported as X.

I or X indicates whether internal microprocessor memory or external memory is to be addressed.

Addr is a 16-bit memory address.

Value is the a byte value in the range 2 to 255 to be overwritten at memory location Addr on a Set or the number of bytes to be reported back to the host on a Get.

Chapter 4: RS-232 serial communication

Signals, timing, character codes, etc.

SmartSet adaptation to RS-232

Hardware Handshaking

The ESAU controller recognizes hardware handshake signals typically implemented in EIA RS-232 communications. In this type of flow control, the host receives data generated by the controller until the host's available resources (buffer space or processing time) limit the flow. At this point the host can turn off the flow of data from the controller by asserting one of the handshake signals (DTR or RTS).

When the controller receives a valid command, it asserts the handshaking signal CTS. This allows a host conforming to the EIA RS-232 to properly suppress further output. When the command has been processed, the controller negates this handshaking signal. The signal DSR is employed as a "channel valid" indicator. It is connected directly to the controller's power supply. Thus, this signal indicates that a controller is present and powered on.

SmartSet™ Packet Structure

The basic message unit passed between the host and the ESAU controller is a "packet". A packet includes a "command" sent by the host or a "response" sent by the controller. Additional data are included in a packet to guard against possible corruption of the command over the serial link.

The SmartSet™ Serial Packet

The structure of the standard serial communication packet is:

<Lead-In><Eight-byte Command or Response><Checksum>

The total length for this packet is 10 bytes. The components of the packet are described in the

following paragraphs.

The "Lead-In" is a character which is used to signal the start of a packet. For the standard serial packet, the lead-in is the ASCII character 'U' (55h). This value was chosen due to its distinct alternating bit pattern.

The command or response format is described in Section “**Error! Reference source not found.**” on page **Error! Bookmark not defined.**

The trailing checksum byte is a value which may be used to check the validity of the communication. It is calculated as follows:

Checksum = AAh + Lead-In + 8 Data Bytes

where the addition retains only the least significant byte of the sum. The SmartSet mode binary response packet format always contains this calculated checksum value. Other response formats do not. The host is generally not required to send a properly calculated value (although a dummy value is required to provide the correct packet length).

Chapter 5: USB communication

Signals, tokens, packets, etc.
SmartSet adaptation to USB

Appendix A: (style ChapterTitle)

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

Glossary (style GlossaryTitle)

The even page header for this section is NOT linked to the previous section. This keeps the header from displaying the last previous chapter title.

Table 4 Terms and their definitions

<i>term</i>	The meaning for “term” appears here.
<i>another term</i>	The meaning for “another term” appears here...
<i>etc.</i>	...and so on.

Bibliography (style BiblioTitle)

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

Index