

**Lucas Control Systems Products  
Deeco™ Systems**

**Resistive Touch Controller  
Model LDC 1000  
USER MANUAL**

**Manual P/N: 13420  
Manual Revision: 5.0  
Manual Revision Date: June 1996**

Copyright 1996, Lucas Automation and Control Engineering, Inc. (L.A.C.E., Inc.)

All rights reserved. Deeco is a registered trademark of LACE, Inc. All other trademarks are the property of their respective owners. Information furnished by Deeco Systems is believed to be accurate and reliable. However, no responsibility is assumed by Deeco Systems for its use, nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent rights of LACE, Inc.

Printed in the USA.



## **How to contact Deeco Systems for sales or technical support:**

**LCSP, Deeco Systems**

**31047 Genstar Road**

**Hayward, California USA 94544-7831**

**Phone: 1-800-376-1154 or 510-471-4700**

**FAX: 510-489-3500**

**Technical Support E-Mail: 76325,3043 or 76325.3043@COMPUSERVE.COM**

**Internet: <http://www.golucas.com>**

**Technical Support BBS: 510-471-5402**

Baud Rate:	up to 14400 Baud
Data Bits:	8
Parity:	None
Stop Bits:	1
Flow Control:	XON/OFF
Emulation:	ANSI

**For Returns** - Contact Deeco Systems' customer service for a Return Authorization (RA) number prior to shipping product to the factory. Freight to the factory is prepaid by the customer. Freight return to the customer is paid by Deeco Systems. Ship product in its original packaging or equivalent to prevent transit damage.

## **Corporate Headquarters**

**Lucas Control Systems Products**

**1000 Lucas Way**

**Hampton, VA 23666**

**Phone: 804-766-1500**

**FAX: 804-766-3979**

**Deeco™ Systems and Duralith™ Products are a division of Lucas Control Systems Products.**



# Table of Contents

<b>1.0 INTRODUCTION .....</b>	<b>1</b>
<b>2.0 TOUCH SCREENS .....</b>	<b>3</b>
2.1 OPERATION .....	5
<b>3.0 CONTROLLER HARDWARE.....</b>	<b>9</b>
3.1 CONTROLLER FEATURES.....	9
3.2 BOARD LAYOUT.....	10
3.3 JUMPER CONFIGURATION.....	10
3.4 POWER INPUT.....	11
3.5 HOST CONNECTION .....	12
3.6 TOUCH SCREEN INTERFACE.....	13
<b>4.0 CONTROLLER INTERFACE .....</b>	<b>15</b>
4.1 HOST COMMUNICATION PROTOCOL.....	15
4.2 COMMAND CONVENTION .....	15
4.3 COMMAND SET DESCRIPTION.....	15
<i>Command 00H: Null Command (^@)</i> .....	15
<i>Command 01H: Self Test (^A)</i> .....	16
<i>Command 02H: Time Out (^B)</i> .....	16
<i>Command 03H: Baud Rate (^C)</i> .....	17
<i>Command 04H: End of String (^D)</i> .....	17
<i>Command 05H: Enable/Disable Serial Reset (^E)</i> .....	17
<i>Command 06H: Serial Port Mode (^F)</i> .....	18
<i>Command 08H: Sensitivity (^H)</i> .....	18
<i>Command 0AH: Set IR Report Mode (^J)</i> .....	19
<i>Command 0CH: Report Baud Rate (^L)</i> .....	19
<i>Command 0DH: Report Serial Mode (^M)</i> .....	20
<i>Command 12H: Report Touch Type (^R)</i> .....	20
<i>Command 15H: POST results (^U)</i> .....	20
<i>Command 16H: Version (^V)</i> .....	21
<i>Command 17H: Report Sensitivity (^W)</i> .....	21
<i>Command 18H: Touch Mode (^X)</i> .....	21
<i>Command 1AH: Configure (^Z)</i> .....	22
<i>Command 1BH: Configuration Report (^I)</i> .....	22

4.4 UNSOLICITED REPORT LIST .....	22
<i>Entry:</i> .....	22
<i>Track:</i> .....	23
<i>Exit:</i> .....	23
<b>5.0 MOUSE EMULATION ON PC COMPATIBLE SYSTEMS .....</b>	<b>25</b>
5.1 INTRODUCTION.....	25
5.2 INSTALLATION - DOS MOUSE.....	26
5.3 INSTALLATION - WINDOWS MOUSE .....	27
5.4 OPERATION .....	27
<b>6.0 SETUP AND DEBUG UTILITIES .....</b>	<b>29</b>
6.1 RT_CFG.EXE: BASIC CONFIGURATION AND TOUCH MONITORING PROGRAM .....	30
6.2 HOST.EXE - A RAW-MODE TERMINAL EMULATOR.....	34
<i>Command-line switches:</i> .....	34
<i>Environment variable:</i> .....	34
<i>Using HOST</i> .....	35
<b>APPENDIX A: LINEARITY OF LUCAS TOUCH SCREEN.....</b>	<b>37</b>
<b>APPENDIX B: CONTROLLER TECHNICAL INFORMATION.....</b>	<b>43</b>
<b>INDEX .....</b>	<b>45</b>

## 1.0 INTRODUCTION

The LDC1000 Touch Screen Controller has been developed for use with the family of Analog Resistive Touch Screens produced by Lucas Control Systems Products MMI Division. This manual will give a detailed description of the commands by which a user can control and communicate with the LDC1000.

In addition to the basic Controller, this manual also contains details of the Touch-Mouse drivers which have been developed for use with the LDC1000 Controller in a PC-DOS or Windows™ environment. These will only work with a fully PC compatible system. These drivers provide an interface to the LDC1000 which emulates a mouse and allows the Touch Screen to work directly with application programs which can use a mouse input.

Users who are writing new application programs will probably prefer to interface directly with the LDC1000, but for those who already have mouse compatible application programs the use of the Touch-Mouse drivers in conjunction with the LDC1000 is a much easier implementation.

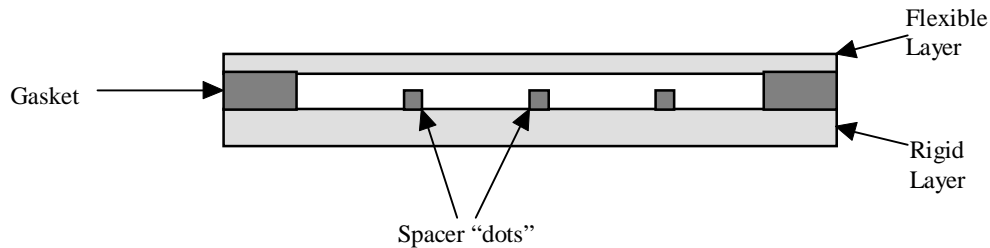
This manual will give a detailed description of the theory and practice of operation of the Controller in later chapters, but first it makes sense to present a discussion of the basics of Resistive Touch Screens.



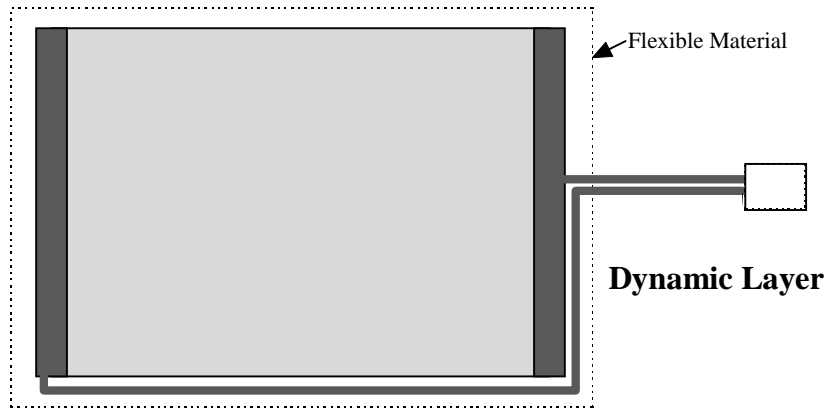
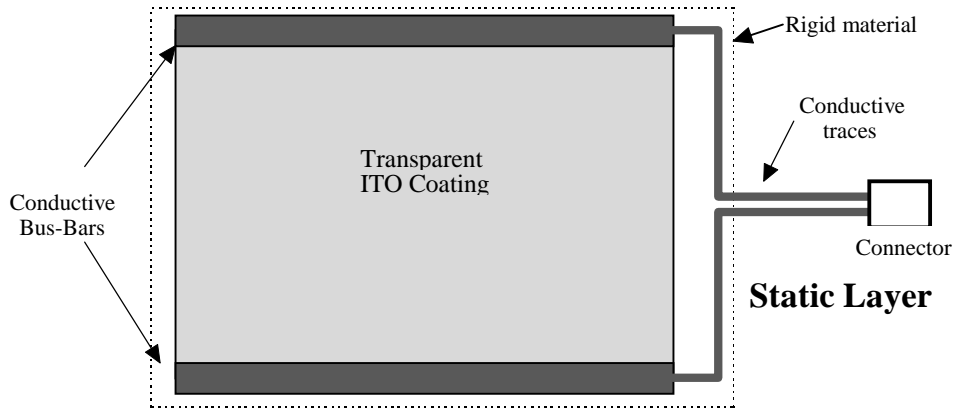
## 2.0 TOUCH SCREENS

As the particulars of the controller itself are discussed, a brief description of the operation of Duralith™ Resistive touch screens is necessary. The analog touch screen is a four-wire resistive type device that has two membranes of closely spaced polyester material. These membranes are made conductive on their facing surfaces by a transparent Indium Tin Oxide (ITO) coating.

An analog touch screen is made from a sandwich of two layers of transparent material. Each layer has a rectangular, resistive coating of Indium Tin Oxide (ITO). The bottom layer, known as the “Static” layer, is rigid. The top or “Dynamic” layer is made of a flexible material. The two layers are assembled together with their ITO coatings facing each other. They are separated by a non-conductive gasket around the outside edge and by small non-conductive “dots” across the “clear view” area.

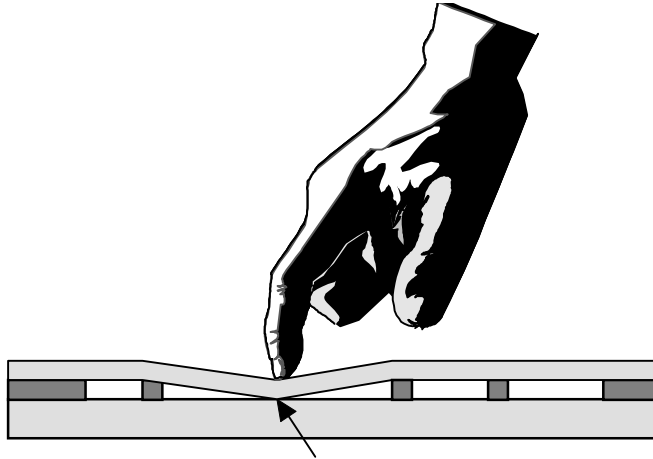


At opposite ends of each layer, metallic bus-bars make electrical contact with the ITO coating and connect it to traces which run back to the touch screen connector. When assembled, the bus-bars of one layer are at right angles to the bus-bars of the other.



## 2.1 Operation

Touching the flexible “Dynamic” face of the touch screen closes the gap between the facing ITO layers and forms an electrical connection.



“Touch” forms electrical contact

A touch screen controller circuit applies a drive voltage across a pair of bus bars. This produces a voltage across the ITO coating that varies in proportion to the distance from each bus bar.



When the screen is touched, the controller senses this voltage using the top layer as an electrode. From this, the controller can calculate the “X” (or “Y”) position. The

## **Resistive Touch Controller -- Model LDC 1000**

---

controller then applies the drive across the other pair of bus bars and repeats the process to obtain the other screen touch dimension.

When a touch is made, the LDC 1000 controller determines a touch location by applying a voltage across the dynamic (top and nearest to the user) membrane while it samples the voltage drop at the touchpoint on the static membrane (bottom and furthest away from the user). The controller automatically changes the “drive” and “sense” connections to obtain both “X” and “Y” voltages.

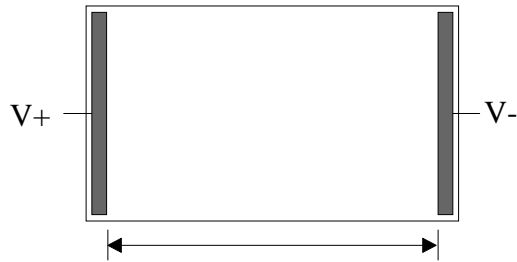
The LDC 1000 controller then digitizes the voltage values and scales them to fit the size of the touch screen based on a calibration procedure. The coordinates of the touch location are sent via a cable to the host computer or microprocessor as a packet of ASCII characters.

The LDC 1000 Touch Screen Controller is a complex, yet simple to operate, proprietary Lucas electronic device built around a high-tech microcontroller IC, which has its own internal EEPROM, 8-bit A/D converter, and an RS-232 serial port. The host system tells the LDC 1000 controller what type of function is desired through the serial interface port. The user selected function can be a single touch location, or a complex stream of touch location information as the user’s finger is moved across the touch screen.

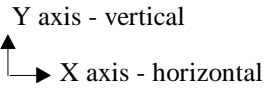
A unique feature of the LDC 1000 touch screen controller is its calibration method. The controller is put into calibration mode by receiving a command sent from the host system.

The user can then exactly calibrate or “fit” the touch screen to the video display behind it by touching the screen along the borders of the active display area. The extreme values thus sensed are placed into the memory of the EEPROM. The values reside there permanently unless they are intentionally changed as a result of another calibration operation (Refer to the Description of the Calibration Mode).

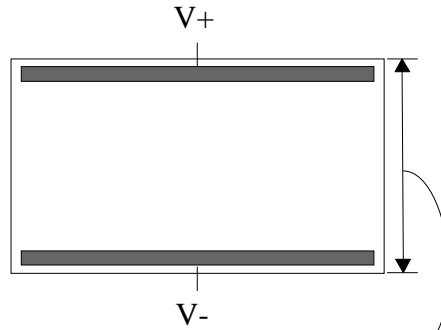
Touch Detection - Horizontal



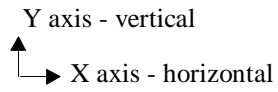
Measure voltage along this axis to determine touch (X) position



Touch Detection - Vertical



Measure voltage along this axis to determine touch (Y) position





## 3.0 CONTROLLER HARDWARE

### 3.1 *Controller Features*

The LCSP Deeco Systems' designed LDC 1000 Resistive Touch Controller is an 80C51 based embedded microcontroller system that allows the user to interface a Lucas Analog Resistive Touch Screen to any computer system via an RS232 serial interface.

- The unit employs an 8-bit ADC that allows for 256 x 256 touch point resolution.
- Calibration mode can adjust the Active Touch Area of the Touch Screen to fit any display size smaller than the Touch Screen itself.
- Mouse emulation. The controller is compatible with existing LCSP Deeco Touch Mouse drivers which allow the Touch Screen to emulate a mouse.
- Baud rate for communication with the host can be 1200, 2400, 4800 or 9600 baud.
- Two command sets built in. The Controller will respond to either the Deeco Touch Controller command set or the Duralith Touch Controller command set. The selection is made via a jumper setting on the board.

### 3.2 Board Layout

The diagram below shows the general layout of the LDC 1000 Resistive Touch Controller with the location of key connectors and components.

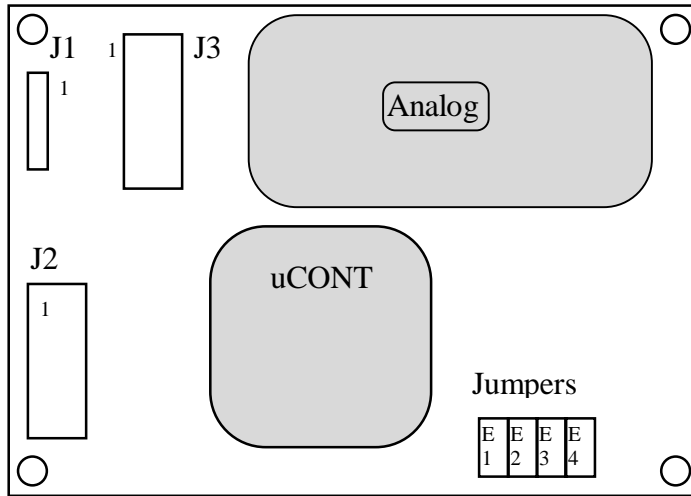


Fig. 1: Board Layout

### 3.3 Jumper Configuration

The LDC 1000 is initially configured via a set of jumpers on the board. Configuration of the board is listed in the table below, where a "0" indicates no jumper, a "1" indicates that a jumper is installed, and an "X" indicates a don't care situation.

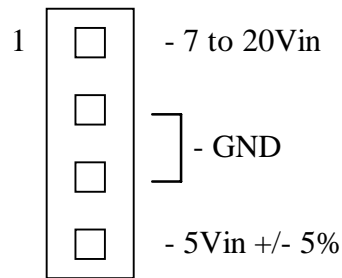
JUMPER SETTINGS				
E4	E3	E2	E1	Function of setting
X	X	X	0	Deeco command set emulation
X	X	X	1	MMI command set emulation
X	0	0	X	1200 Baud
X	0	1	X	2400 Baud
X	1	0	X	4800 Baud
X	1	1	X	9600 Baud
0/1	X	X	X	Jumper E4 is unused

**Table 1: Jumper Configuration**

For proper Deeco Touch Driver operation, the default jumper settings for the touch controller is with no jumpers installed (Deeco command set, 1200 baud). The 1200 baud rate setting is required for proper mouse emulation. When using the Deeco Touch Driver, the baud rate will be changed to a higher rate (4800 or 9600) via the command protocol.

### 3.4 Power Input

Connector J1 is used to connect power to the controller card. Either +6V to +18V can be supplied on pin 1, or a +5V rail can be applied to pin 4. Pins 2 and 3 are both ground. Pins 1 and 4 should never be connected to a voltage rail simultaneously.



**Fig. 2: Power Connector J1**

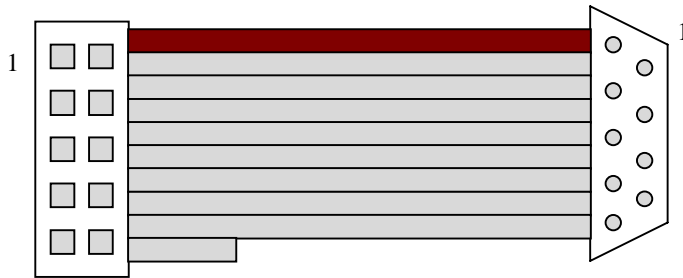
### 3.5 Host Connection

Connector J2 is used for serial RS232 communications with the host, and is laid out to be compatible with a 9-pin IDC DSUB connector. The Resistive Touch Controller acts as a DCE device, much like a modem or mouse, which means that it uses a female DSUB connector. The direction of the signals are relative to the host DTE device, so that a signal that goes "out" originates with the host, and "in" means it is sent to the host.

No.	9-pin DSUB	Header J2	Signal Description	Direction
1	1	1	DCD - Carrier Detect	in
2	2	3	RxD - Receive Data	in
3	3	5	TxD - Transmit Data	out
4	4	7	DTR - Data Terminal	out
5	5	9	Ready	--
6	6	2	GND - Signal Ground	in
7	7	4	DSR - Data Set Ready	out
8	8	6	RTS - Ready to Send	in
9	9	8	CTS - Clear To Send	in
10	--	10	RI - Ring Indicator No Connection	--

**Table 2: RS232 Communications Connector Pin-out**

The easiest way to connect the Resistive Touch Controller to a PC is to use a 10-pin IDC connector with 10 wire flat ribbon cable, which is connected to a female 9-pin DSUB IDC connector, pin 1 to pin 1. A null modem is not required for proper operation of the unit.



**Fig. 3: Header to DSUB connection**

The touch controller serial channel can be reset to the default values by toggling the RTS/DTR serial lines together; i.e., both RTS from low to high, and DTR from high to low for 55 mS. The ability to reset via the RTS/DTR serial lines can be disabled through Command 05H in the Deeco Command Set. Before resetting the serial channel, the host and controller should be in an idle state. The touch controller will respond to the serial reset by transmitting the byte <4DH>, 'M', to the host.

### 3.6 Touch Screen Interface

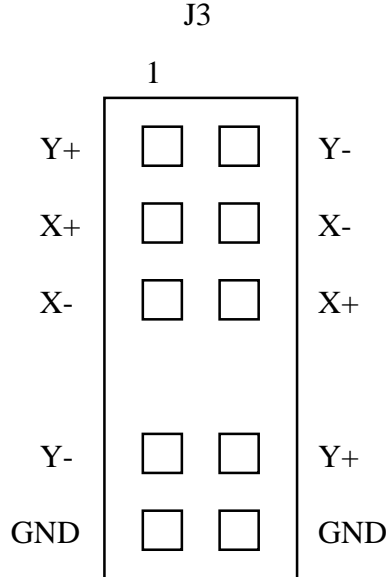


Fig. 4: Touch Screen Interface

## **Resistive Touch Controller -- Model LDC 1000**

---

The interface to the resistive touch screen is made via connector J3. Support is provided for both right side and left side mounting screens. The pins are defined as follows:

1, 10	- Y+ drive
3, 6	- X+ drive
5, 4	- X- drive
7, 8	- key
9, 2	- Y- drive
11, 12	- Chassis ground for EMI Shield

Pins 1, 3, 5, 9 are used for Right Side touch screens (max. coordinates are in lower right of screen).

Pins 2, 4, 6, 10 are used for Left Side touch screens (max. coordinates are in upper left of screen).

Pins 11 and 12 are for screens with an EMI shield. These pins are connected to chassis ground.

Pins 7 and 8 are used for keying purposes only.

## 4.0 CONTROLLER INTERFACE

### 4.1 Host Communication Protocol

Baud rate is set by jumpers on the controller card, as documented above or by issuing the change baud rate command which is described below. The communication protocols are as listed below.

#### **DEECO emulation mode**

7 Data Bits, No Parity, 2 Stop Bits.

The touch controller requires a total of 10 bits, unused bits should be set high.  
XON/XOFF Software Protocol.

#### **MMI emulation mode**

8 Data Bits, No Parity, 1 Stop Bit.

The touch controller requires a total of 11 bits, unused bits should be set high.  
XON/XOFF Software Protocol.

### 4.2 Command Convention

The spaces in the formats are for clarity only. There should be no spaces, brackets or braces in the commands and parameter lists. Eight ( 8 ) bit binary values are represented in the form <xxH>, where 'xxH' is the binary value in hexadecimal form. ASCII text transmissions are enclosed in single quotes, 'xx'.

The following notation is used to represent pre-defined values:

<ESC> is escape character (1BH).

<EOT> is end-of-transmission (04H).

### 4.3 Command Set Description

#### **Command 00H: Null Command (^@)**

The Null command has no action and no response given.

Format: <00H>

Response: None

**Command 01H: Self Test (^A)**

Controller performs self-diagnostic and RAM/ROM tests. Response is issued after command is submitted and completed.

Format: <01H>  
Response: <ESC> S *Pr* <EOT>

*Pr* is the result represented as two ASCII encoded decimal bytes:

- '00': all tests passed
- '01': RAM test failed
- '02': ROM test failed, bad check sum.
- '03': Both RAM and ROM test failed.
- '04': NVRAM read failed.
- '05': RAM and NVRAM test failed.
- '06': ROM and NVRAM test failed.
- '07': RAM, ROM, and NVRAM test failed.

**Command 02H: Time Out (^B)**

Sets the inter-character time out period for parameters being passed in a command.

Format: <02H> *Phi Plo*  
Parameters: *Phi* and *Plo* are 8 bit binary values used to set the inter-character time out period for commands utilizing parameters. The timer unit is in 0.1 milli-seconds, generated by multiplying *Phi* and *Plo*. A value of *Phi* = 1 and *Plo* = 1 sets the inter-character time out period to 0.1 milli-seconds.  
Response: None

Example: <02H> <3CH> <0AH> or (^B<^J)  
To set a time out period of 60 milli-seconds, set *Phi* to 60 and *Plo* to 10.

**NOTE:** *The default time out period is 500 milli-seconds. Phi should not be set to zero.*

**Command 03H: Baud Rate (^C)**

Sets the touch controller's serial port baud rate. No response is given. Host must change its serial port controller's baud rate to continue communications with the touch controller.

Format: <03H> *Pbr*

Parameters: *Pbr* is the requested baud rate, 8 bit binary value:

<00H> : 19200

<01H> : 9600

<02H> : 4800

<03H> : 2400

<04H> : 1200

<05H> : 600

<06H> : 300

Response: None

Default: 1200

**Command 04H: End of String (^D)**

Signal the end of host string.

Format: <04H>

Response: None

**Command 05H: Enable/Disable Serial Reset (^E)**

Command enables or disables the RTS/DTR serial reset used for mouse emulation.

Format: <05H> *Pset*

Parameter: *Pset* is a 8 bit binary value and enables/disables the ability to reset the controller serial channel via the RTS/DTR serial lines. A non-zero value enables the reset option, a zero value disables the reset option.

Response: None

**Command 06H: Serial Port Mode (^F)**

Sets the communication mode of controller's serial port.

Format: <06H> *Pmode*

Parameters: *Pmode* is the requested mode, 8 bit binary value.

<00H> : 7 data bit, no parity, 2 stops - DEFAULT

<01H> : 7 data bit, odd parity, 1 stop

<02H> : 7 data bit, even parity, 1 stop

<03H> : 7 data bit, no parity, 2 stops

<04H> : 8 data bit, no parity, 1 stop

<05H> : 8 data bit, odd parity, 1 stop

<06H> : 8 data bit, even parity, 1 stop

<07H> : 8 data bit, no parity, 2 stops

Response: None

Example: <06H> <04H> or (^F^D)

Sets the serial port mode to 8, none, and 1.

**NOTE:** *Modes 0-3 require that the host sends 10 data bits, modes 4-7 require that the host sends 11 data bits, including the start bit. Unused bits must be set high.*

**Command 08H: Sensitivity (^H)**

Sets the sensitivity of the touch scanner. Value is stored in NVRAM and retrieved at power-up.

Format: <08H> *Px Py*

Parameters: Both of the parameters are 8 bit binary values. *Px* is the maximum amount that two consecutive readings can differ by in the horizontal direction and still be considered one touch point. *Py* is the maximum amount that two consecutive readings can differ by in the vertical direction and still be considered one touch point. Smaller numbers indicate higher sensitivity.

Default values are *Px* = 03 and *Py* = 03.

Response: None. See Command 17H for data retrieval.

Example: <08H> <04H> <01H> or (^H^D^A)

Sets the sensitivity to 4 in the x-direction, and 1 in the y-direction.

**Command 0AH: Set IR Report Mode (^J)**

Enables or disables the different reporting modes for Deeco Touch Controller emulation.

Format: <0AH> Pmode

Parameters: Pmode is the requested mode, 8-bit binary value, and is a bit mask used to enable or disable touch event reports. Pmode's bits have the following definitions:

Bit	Event	Definition
0	Entry	Set enables sending of entry reports Cleared disables sending of entry reports
1	Track	Set enables sending of track reports Cleared disables sending of track reports
2	Exit	Set enables sending of exit reports Cleared disables sending of exit reports

**NOTE:** *Pmode default state at power up is <07H>, which means all touch events are reported.*

Response: None.

Example: To enable the reporting of both entry and exit reports, and disable the reporting of track reports, send the following message:

<0AH> <05H> or (^J^E)

**Command 0CH: Report Baud Rate (^L)**

Report controller serial port baud rate

Format: <0CH>

Response: <ESC> D *Pbr* <EOT>

*Pbr* is the baud rate representation in two ASCII encoded decimal bytes:

'00' : 19200

'01' : 9600

'02' : 4800

'03' : 2400

'04' : 1200

'05' : 600

'06' : 300

**Command 0DH: Report Serial Mode (^M)**

Reports the current communication mode of the serial channel.

Format: <0DH>

Response: <ESC> N *Pmode* <EOT>

*Pmode* is the serial mode, which is represented by two ASCII encoded decimal bytes:

- '00' : 7 data bit, no parity, 2 stop
- '01' : 7 data bit, odd parity, 1 stop
- '02' : 7 data bit, even parity, 1 stop
- '03' : 7 data bit, no parity, 2 stops
- '04' : 8 data bit, no parity, 1 stop
- '05' : 8 data bit, odd parity, 1 stop
- '06' : 8 data bit, even parity, 1 stop
- '07' : 8 data bit, no parity, 2 stops

**Command 12H: Report Touch Type (^R)**

Reports the type of touch controller installed. Command is used to differentiate between IR and Resistive Touch Controllers.

Format: <12H>

Response: <ESC> I *Ptype* <EOT>

*Ptype* is the touch type in use and is one of the following:

- <49H> Touch type is IR Touch.
- <52H> Touch type is Resistive Touch.

**Command 15H: POST results (^U)**

Report Power On Self Test ( POST ) results.

Format: <15H>

Response: <ESC> S *Pr* <EOT>

*Pr* is the result represented as two ASCII encoded decimal bytes:

- '00': all tests passed
- '01': RAM test failed
- '02': ROM test failed, bad check sum.
- '03': Both RAM and ROM test failed.
- '04': NVRAM read failed.
- '05': RAM and NVRAM test failed.
- '06': ROM and NVRAM test failed.
- '07': RAM, ROM, and NVRAM test failed.

**Command 16H: Version (^V)**

Reports version of embedded software.

Format: <16H>

Response: <ESC> V Pv Pr <EOT>

*Pv* is the version in one byte ASCII encoded decimal.

*Pr* is the revision in one byte ASCII encoded decimal.

**Command 17H: Report Sensitivity (^W)**

Report touch scanner sensitivity.

Format: <17H>

Response: <ESC> J Px Py <EOT>

*Px* is the maximum amount that consecutive readings can differ by in the horizontal direction and still be considered one touch point.

*Py* is the maximum amount that consecutive readings can differ by in the vertical direction and still be considered one touch point.

Default values are 04 and 04.

**Command 18H: Touch Mode (^X)**

Reports the enabled touch report modes for Deeco emulation.

Format: <18H>

Response: <ESC> R Pmode <EOT>

*Pmode* is the current touch-reporting mode, in two ASCII encoded hexadecimal characters, and represents a bit mask where the bits are used to indicate the enabling or disabling of touch event reports. *Pmode*'s bits have the following definition:

Bit	Event	Definition
0	Entry	Entry reports are enabled (1) or disabled (0)
1	Track	Tracking reports are enabled (1) or disabled (0)
2	Exit	Exiting reports are enabled (1) or disabled (0)

Example: <ESC> 'R' '0' '5' <EOT>

Indicates that the entry and exit touch events are enabled, while the track touch reports are disabled.

**Command 1AH: Configure (^Z)**

Enters screen configuration mode wherein the active area of the touch screen can be interactively defined by the user. This mode is exited by the issuing of the Configure command a second time.

Format: <1AH>

Response: <ESC> C Xmin Ymin Xmax Ymax <EOT>

*Xmin, Ymin, Xmax, Ymax* are in two-digit ASCII HEX representation

Example: If the current screen boundaries are (2, 6),(252, 249), then the response will be:

<ESC> 'C' '0' '2' '0' '6' 'F' 'C' 'F' '9' <EOT>

**Command 1BH: Configuration Report (^[])**

Reports the screen's current configuration values. These values represent the minimum and maximum physical values that will be translated to logical values. Reports outside of the submitted range will be ignored.

Format: <1BH>

Response: <ESC> C Xmin Ymin Xmax Ymax <EOT>

*Xmin, Ymin, Xmax, Ymax* are in two-digit ASCII HEX representation

Example: If the current screen boundaries are (2, 6),(252, 249), then the response will be:

<ESC> 'C' '0' '2' '0' '6' 'F' 'C' 'F' '9' <EOT>

## **4.4 Unsolicited Report List**

These are reports sent to the host without a command request from the host. These reports represent interaction on the touch screen.

**Entry:**

New touch detected

Format: <ESC> E Py Px <EOT>

*Py, Px* is the touch coordinate at entry time in ASCII HEX (two bytes each).

Example: Touch detected at coordinate (43, 19)

<ESC> 'E' '1' '9' '4' '3' <EOT>

**Track:**

Touch has moved to new position.

Format: <ESC> T Py Px <EOT>

*Py, Px* are new touch coordinate in ASCII HEX (two bytes each).

Example: Touch has moved to coordinate (38, 46)

<ESC> 'T' '4' '6' '3' '8' <EOT>

**Exit:**

Touch has exited the bezel.

Format: <ESC> X <EOT>



## 5.0 MOUSE EMULATION ON PC COMPATIBLE SYSTEMS

### 5.1 Introduction

The Lucas touch controllers provide an integrated touch input system which can be used to provide a simple, intuitive operator interface. The touch system can be configured to emulate the standard AT mouse input device. All AT compatible programs which use the mouse as an input device can use the touch system as well.

Like the standard mouse, the touch mouse provides two software drivers to interface applications to the physical device. One driver is suitable for MS-DOS based applications and the other is used for Microsoft® Windows applications. The drivers process touch reports received from the touch system on COM channels and translates them into mouse compatible format for use by applications. The translation is transparent to applications, such that they need not know that the mouse has been replaced by the touch system.

The mouse emulation software drivers consist of three files:

- 1) MOUSE.COM
- 2) LDTOUCH.COM
- 3) LWTOUCH.DRV

The files MOUSE.COM and LDTOUCH.COM together comprise the MS-DOS touch mouse driver. These two files should be copied to the same directory. The directory name can be your choice, though "\MOUSE" is suggested. This directory may be specified in the "PATH" statement of the autoexec.bat file to gain access to the files.

LWTOUCH.DRV is the Microsoft Windows based touch driver. It should be copied to the \system sub directory of the Windows installation. For example, if Microsoft Windows were installed on C drive in the directory \windows, LWTOUCH.DRV should be copied to C:\windows\system.

## **5.2 Installation - DOS Mouse**

The MS-DOS based driver consists of two parts: the Microsoft mouse driver and the Lucas touch driver. Both must be loaded for proper mouse emulation. The Microsoft driver is loaded first. The Lucas touch driver is loaded second. It provides the link between the touch system and the Microsoft driver.

To load the modified Microsoft driver use the command:

```
C:>mouse
```

This format assumes that the directory where MOUSE.COM and LDTOUCH.COM were installed is accessible via the DOS path. If this is not the case, the full path should be specified such as:

```
C:>\mouse\mouse
```

The driver will install and report success. Next, the Lucas touch driver, LDTOUCH.COM, should be installed. It is installed in a similar fashion the MOUSE.COM:

```
C:>ldtouch
```

or

```
C:>\mouse\ldtouch
```

The Lucas driver will not load if the mouse driver is not loaded or the touch system is not working.

***NOTE: The use of the Microsoft mouse driver as the application interface to the touch system mouse emulation insures compatibility with true mouse operation.***

### 5.3 Installation - Windows Mouse

The Windows driver, LWTOUCH.DRV, must first be copied into the \SYSTEM sub directory of the Windows installation. If Windows exists on drive C: in the \WINDOWS sub directory, then the driver must be copied to C:\WINDOWS\SYSTEM. To specify the Lucas driver, the Windows initialization file, SYSTEM.INI, must be modified. The file may be modified by any suitable text file editor, such as the MS-DOS 5.0 EDIT editor. The mouse driver is specified by a line in SYSTEM.INI which looks like this:

```
mouse.driv=mouse.driv
```

This particular line would specify the Microsoft mouse windows driver, supplied with Windows. To specify the Lucas driver, change the line to:

```
mouse.driv=lwtouch.driv
```

No other changes are necessary. When Windows is started the driver will load. If the touch system is not operational Windows will behave exactly as if the standard mouse driver were loaded without a mouse present. This behavior is characterized by the absence of the mouse cursor. Windows may still be controlled from the keyboard. If present, a Microsoft compatible mouse can work in conjunction with the Lucas Windows Touch Drivers.

### 5.4 Operation

Both the MS-DOS and Windows touch mouse drivers emulate a one-button mouse. They simulate mouse button actuation and motion in a similar manner. There are three significant touch "events" used to emulate a mouse. The first event is the "entry" event. An entry is the first touch by a finger or stylus detected by the system. "Track" events are changes in the stylus position after the entry event. An "exit" event is the removal of the stylus from the touch system.

When the touch system detects an entry touch event, two mouse events are generated. First the mouse position is updated to be identical with the touch position. This is characterized by the visible cursor moving to the position beneath the stylus. Second, the left mouse button is reported to be depressed. During subsequent track events, the mouse position is continually updated. An exit event generates a left button released mouse event.

## **Resistive Touch Controller -- Model LDC 1000**

---

This method of emulation is based on the simple rules:

- 1) the mouse position is identical to the stylus location
- 2) as long as the stylus is present the left button is "pressed"

This allows operation such as click and drag to be accomplished quite easily.

A special "double-click" algorithm has been developed to insure that double-click operations, a rapid entry-exit-entry-exit, are cleanly detected and reported.

## 6.0 SETUP AND DEBUG UTILITIES

The Resistive Touch controller, when configured for Deeco emulation, supports the full range of Deeco touch-controller commands. (For a full list of these commands, see the file RESTOUCH.COMD included on this disk). In order to assist users with accessing the controller, Deeco provides two utility programs:

### RT\_CFG.EXE

This MSDOS program has a user-friendly interface and provides access to basic functionality of the unit, including configuring the active area of the screen, and displaying any touch reports which are received from the controller.

### HOST.EXE

This MSDOS program has a very rudimentary user interface, but permits direct transmission of any commands to the controller. Any data which is received from the controller is displayed without modification on the host display. HOST can be used to send Deeco commands which are not supported by RT\_CFG, and to observe the target's responses.

These two utilities are documented fully below.

## 6.1 RT\_CFG.EXE: Basic configuration and touch monitoring program

### Command-line switches:

***-c[commport:baud,parity,databits,stopbits]***

This permits specifying the COM port and parameters which are used by RT\_CFG. The default is COM1:9600,N,8,1(port COM1, 9600 baud, no parity, 8 data bits, 1 stop bit). In order to change ANY ONE of these parameters, the entire argument list must be specified.

***-m***

This specifies that the MMI command set should be used. The default is to support the Deeco command set. The Deeco command set is documented in the file RESTOUCH.CMD. For a list of MMI emulation commands, see the relevant MMI Analog Touch Screen Controller documentation.

***Environment variable:***

To simplify the task of selecting these parameters for other standard settings, the RT\_CFG environment variable is supported. If this variable is created, the program will use its values rather than the default values. For example, to set the serial port to COM2, at the MSDOS prompt type

```
set RT_CFG=-ccom2:9600,n,8,1
```

before running RT\_CFG. It will then use the values from the environment as its defaults. This statement could be placed in AUTOEXEC.BAT to make these settings permanent.

If RT\_CFG is not found in the environment, the HOST environment variable will be used if present. It has the same format as RT\_CFG.

***NOTE: The arguments on the command line will take precedence over those in the environment variable.***

## Using RT\_CFG

When the program is first loaded, it is in either the IDLE state (if emulating MMI command set) or waiting for touch reports (if emulating Deeco command set). No commands are initially sent to the controller. The program will display a logo and Deeco copyright, the current serial port parameters, and the current emulation mode (Deeco or MMI). This can be used to confirm that the program accepted any setup parameters that the user specified.

In the center of the screen is the current program status, and below that is a menu of keystrokes for the current emulation mode.

### *Touch report format*

Incoming touch reports are displayed in this format:

```
Entry point = (114,122)
Tracking point = (115,124)
Exit report was received
```

### *Menus for DEECO emulation*

When RT\_CFG is emulating the Deeco command set, it supports the following commands:

C = Begin/End Configuration process

This sends a "begin configuration" command to the Touch controller, then continuously displays the current active screen boundaries as reported by the controller. After sending this command, the user should begin touching screen points which define the desired area for the active screen. RT\_CFG will display messages which look like this:

```
Xmin = 058, Ymin = 053
Xmax = 196, Ymax = 192
status = CONFIGURATION monitoring
```

You should continue moving the touch point around the active area of the screen until you no longer see changes in the boundary positions.

Once you press "C" again (or "Escape") in RT\_CFG, the status message will change to "idle", and the current boundary values will be written into non-volatile memory on the controller.

S = Read the current screen boundary values from the non-volatile memory and display them on the screen.

T = Set new "Touch Report" delay time.  
When a new value is sent to the controller, it is written into the non-volatile memory and takes effect immediately. RT\_CFG will then read the value back from the controller and display it on the screen, to confirm that it was received successfully.

Esc = Exit from RT\_CFG and return to MSDOS.

***Menus for MMI emulation***

When RT\_CFG is emulating the MMI command set, it supports the following commands:

- 1 = Place touch controller in "single-touch" mode, then display incoming touch reports as they are received.  
Pressing "Escape" will return RT\_CFG to idle status.
- 2 = Place touch controller in "streaming" mode, then display incoming touch reports as they are received.  
Once a "release" report is received in streaming mode, RT\_CFG will place the controller in "standby" mode automatically, then return to idle status.

The MMI-emulation touch displays look like this:

Touch point = (114,122)  
Release point = (115,124)  
status = waiting for touch reports

C = Begin/End Configuration process  
This sends a "begin configuration" command to the Touch controller. In MMI emulation, only TWO points may be touched, unlike Deeco mode where as many samples may be provided as the user wishes. Once the "begin configuration" command is sent, RT\_CFG will prompt the user for two touch points, and will then display the resulting screen boundaries, in the same format as shown in the discussion of Deeco emulation, above.

In MMI emulation mode, the user will not need to press "C" a second time to terminate configuration.

S = Read the current screen boundary values from the non-volatile memory and display them on the screen.

T = Set new "Touch Report" delay time. When a new value is sent to the controller, it is written into the non-volatile memory and takes effect immediately. RT\_CFG will then read the value back from the controller and display it on the screen, to confirm that it was received successfully.

Esc = Exit from RT\_CFG and return to MSDOS.

## 6.2 HOST.EXE - A raw-mode terminal emulator

### Command-line switches:

-c[commport:baud,parity,databits,stopbits]

This permits specifying the comm. port and parameters which are used by HOST. The default is COM1:9600,N,8,1 (port COM1, 9600 baud, no parity, 8 data bits, 1 stop bit).

**NOTE: In order to change ANY ONE of these parameters, the entire argument list must be specified.**

-x Toggles software handshaking (XON/XOFF).

Default: NO software handshaking

-d[delay]

Sets inter-character delay, specified in milliseconds. This is the amount of time waited between sending characters to the target. It does not effect the received-data stream.

Default: 0 milliseconds delay

### Environment variable:

To simplify the task of selecting these parameters for other standard settings, the HOST environment variable is supported. If this variable is created, the program will use its values rather than the default values. For example, to set the serial port to COM2, at the MSDOS prompt type:

```
set HOST=-ccom2:9600,n,8,1
```

before running HOST. It will then use the values from the environment as its defaults. This statement could be placed in AUTOEXEC.BAT to make these settings permanent. The arguments on the command line will take precedence over those in the environment variable.

## Using HOST

HOST is similar in functionality to many basic terminal emulator such as Windows Terminal, Procomm Plus, etc. However, each of these other programs "swallows" certain received characters (such as ESCAPE) or prohibits transmission of special characters. HOST, on the other hand, has NO special characters except function keys. ALL characters which are typed on the HOST keyboard are both displayed on the screen (without interpretation) and transmitted to the target. Similarly, EVERY character which is received from the target is displayed as a single ASCII character, without interpretation.

### *HOST screen colors*

Transmitted and received characters are displayed in different colors to aid in distinguishing between them. In addition, other colors are used for differentiating special conditions to the user. The colors used by HOST are:

Green = Transmitted characters.

Bright Red = Received characters.

Yellow = Received characters after XOFF is received.

Cyan = Menus and other HOST messages to user.

Bright Red on Blue background = ERROR messages.

These colors can not be set by the user in the current version.

***Menus in HOST***

The HOST main menu supports these keyboard commands:

F1 = Clear screen/Home cursor.

Send an ASCII ClearScreen command to the target. This also clears the host screen and rewrites the main menu.

F2 = Send text file to target.

This transmits a file from a disk drive to the serial port. The user is prompted for a filename; if <Enter> is pressed without specifying a filename, the most recently sent file is re-sent.

***NOTE: ( regarding text file formats) These must be ASCII text files, with lines ending in CR/LF characters. Any line beginning with a semicolon is treated as a comment and is not transmitted to the target.***

Send the string "Escape c" to the target, which is a Deeco system reset.

F4 = Send CSI to target.

F5 = Send CSI> to target.

F9 = Execute DOS shell.

F10 = EXIT from this program.

Any other characters which are typed, are sent to the target and also echoed to the screen without translation.

## APPENDIX A: LINEARITY OF LUCAS TOUCH SCREEN

The purpose of this section of the User Guide is twofold. The first is to provide a definition of linearity with respect to Lucas Touch Screens. The technical reasons for choosing the definition will be discussed.

Once the “Lucas Linearity Definition” is known by Lucas customers, then they are free to apply their own yardstick for use in their specific application for a touch screen.

The second purpose of this section is to describe the Lucas Touch Screen Linearity Testing Procedure. This is the procedure used by Lucas’ in-house technicians to final inspect each touch screen before it leaves the factory.

Let us first look at the linearity definition.

### *The Lucas Definition of Linearity*

The formula used to determine the linearity error of a Lucas touch screen is:

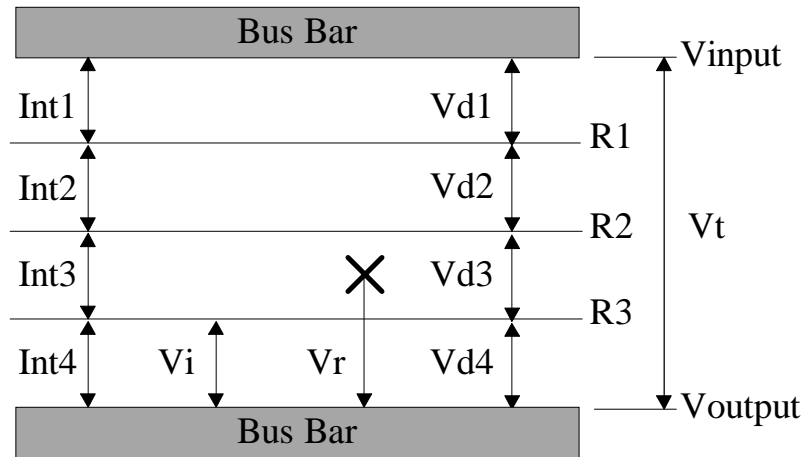
$$\begin{aligned} & \% \text{ Linearity Error} \\ & \text{at a single point} = 1/V_t (V_i - V_r) * 100 \\ & \text{on one axis} \end{aligned}$$

where

$$V_r = V_t - [ V_d (1 - R/N) ]$$

- $V_i$  = An expected touch voltage which is represented by a voltage drop
- $V_t$  = The total voltage applied across the two bus bars =  $V_{input} - V_{output}$
- $V_r$  = The voltage drop which is actually occurring at the touch location being tested
- $R$  = The row in which the touch is being made
- $N$  = The total number of intervals in the text matrix + 1
- $V_d$  = The expected voltage drop for an interval in the case that a screen is perfectly linear
- $V_d$  =  $V_t / \text{The number of intervals} = V_t / [\text{The number of rows} + 1]$

The sketch shown below provides a visual description of some of the definitions shown on the previous page.



$$V_t = Int1 + Int2 + Int3 + Int4$$

$$Int1 = Int2 = Int3 = Int4, \text{ when screen is perfectly linear}$$

When the aspects of the above formulas are applied to a Lucas Touch Screen used in conjunction with the LDC 1000 Controller and its 8-bit microprocessor, some other relationships come into play:

X-Axis error at a single point = The difference between the actual and expected touch location along the X-axis of a touch screen  
256 possible points of resolution in the X direction

Y-Axis error at a single point = The difference between the actual and expected touch location along the Y-axis of a touch screen  
256 possible points of resolution in the Y direction

## *A Closer Look at the Lucas Definition*

As discussed above, by using the LDC 1000 digital controller, we are setting the theoretical number of points that are to be generated by an analog touch screen to a  $256 \times 256 = 65,536$  point total resolution (Because of mechanical and electrical considerations, the actual matrix is in the range of  $220 \times 220$ ). The accuracy with which we can check our linearity will therefore depend on screen resolution.

Example:

Touch Screen panels are typically in the axis size range of 3" to 16" along the X or Y axis.

On a 3" axis panel, physical spacing between possible values will be  $3/256 = 0.01171875$ ". The resolution along the given axis is therefore 0.01171875".

In the case of a 16" axis panel, the physical spacing will be  $16/256 = 0.0625$ ". The resolution along the given axis is therefore 0.0625".

Let us now look at the Lucas method for determining the linearity of a touch screen.

## *The Lucas Method for Determining Linearity of a Touch Screen*

### **Equipment Used**

Each screen is tested by using the LDC 1000 8-bit touch screen controller attached to an IBM PC/AT or compatible. Since the digital controller has a finite resolution, it is the determining factor in the test procedure. As stated above, the 8-bit controller is capable of providing  $256 \times 256$  possible points of grid resolution.

### **The Testing Procedure**

Each touch screen is tested individually. By using the Lucas internal testing program LTSLT (Lucas TouchScreen Linearity Testing), we determine where the test touch locations are to happen. We then use an Asymtek X/Y/Z table, its automatic touching device and the above program to set "touches" at those exact locations. The LTSLT program then drives the Asymtek machine through the programmed series of individual touches on the surface of the screen being tested.

The electrical location of each touch is returned to the program. This location is then compared to the Asymtek programmed mechanical touch location. As a final step, the LTSLT program calculates the linearity error of the screen and assigns a pass/fail rating to the touch screen panel being tested. The results are sent to a line printer. The printout can then be attached to the screen for shipment.

### Linearity Testing Examples

As a first example, a 4" by 4" touch screen is being tested for its linearity. The screen is touched by the user at real coordinates X=1, Y=1. If the Lucas LDC 1000 touch screen controller returns X=64 and Y=64, then the screen is perfectly accurate at that location (It may or may not be "perfectly accurate at other locations"). The "64" is derived by the controller as:

$$256 \times 1 \text{ inch} / 4 \text{ inches} / 64$$

All of the dimensions drop out, and the number 64 simply becomes a relative position. A 64 for the X location and a 64 for the Y location are then sent along to the microprocessor or desktop computer for use.

(If the screen had measured 8" x 8", the X=1, Y=1 would be interpreted by the controller as coordinates 32, 32.)

As a second example, a touch screen is being evaluated by a Lucas technician using the LTSLT program, and the LDC 1000 controller returns X=66, Y=64 in the same situation as stated in the first example above.

$$\text{The X-Axis Error at the point touched} = (66-64) / 256 = 0.00781 * 100 = 0.781\%$$

In actual practice, a Lucas screen is touched numerous places along the X-axis and Y-axis. The X- and Y-Axis Errors are then determined as each point is touched. The error readings are then used by the test software to determine overall screen linearity. The formulas stated earlier in this section are applied. Typically, touches are made every 1/4" to 1" along the axis depending upon the length of the axis.

As the Lucas inspection proceeds, if the average linearity error along the X axis or the Y axis is greater than 5%, then the screen is rejected.

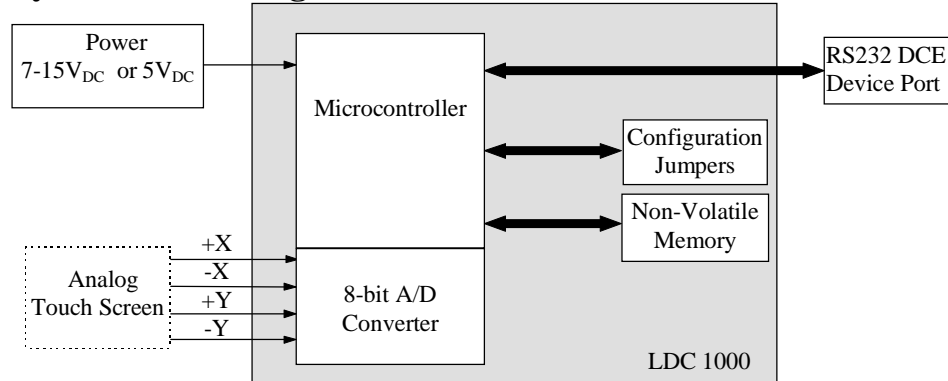
In all tests, if the electrical return from a single point varies more than 10% from its intended location, the panel is rejected.

The Lucas acceptability standard of 5% or less linearity error will normally suffice for most touch screen applications and depends upon screen size. However, screens with a linearity error specification of less than 5% and less than 10% for a single point are available at additional cost.



## APPENDIX B: CONTROLLER TECHNICAL INFORMATION

### System Block Diagram



### Specifications

Supply Voltage	+5V <sub>DC</sub> , +7 to +14V <sub>DC</sub> @ 70°C, or +7 to +16V <sub>DC</sub> @ 45°C
Current draw	50 mA
Touch Point Resolution	8-bit
Data Output	RS232
Baud Rate	1200 - 9600 bits per second (user selectable via jumpers/software)
Data Format	7 data bits, no parity, 2 stop bits Other modes are user selectable via software
Storage Temperature	-55°C to +150°C (-67°F to +302°F)
Operating Temperature	0°C to +70°C (32°F to +158°F)
Size	9.5 cm x 6.0 cm x 1.2 cm (3.74 in x 2.36 in x 0.47 in)



## INDEX

---

### *B*

baud rate, 9, 15, 17, 19, 43  
block diagram, 43

---

### *C*

commands, 6, 9, 10, 11, 15, 16, 22, 26, 30, 31, 32, 34, 36  
controller, 1, 3, 5, 6, 9, 10, 11, 12, 15, 16, 17, 18, 19, 20, 29, 30, 31, 32, 33, 38, 39, 40, 43

---

### *D*

data format, 43  
debug, 29  
display, 6, 9, 29, 31, 32, 33

---

### *H*

hardware, 9  
host, 6, 9, 12, 13, 15, 17, 18, 22, 29, 36

---

### *I*

installation, 25, 26, 27  
interface, 1, 6, 9, 13, 14, 15, 25, 26, 29

---

### *J*

jumper, 9, 10, 11, 15, 43

---

### *L*

linearity, 37, 39, 40, 41

---

***M***

menus, 31, 32, 35, 36

mouse emulation, 9, 11, 17, 25, 26

---

***P***

power input, 11

---

***R***

report, 19, 20, 21, 22, 26, 31, 32, 33

---

***S***

self-test, 16, 20

---

***T***

touch screens, 1, 3, 5, 6, 9, 13, 14, 22, 30, 37, 38, 39, 40, 41