Warranty and Repair Policy Statement

General
Seller warrants that its products furnished hereunder will, at the time of delivery, be free from defects in material and workmanship and will conform to Seller's published specifications applicable at the time of sale. Seller's obligation or liability to Buyer for products which do not conform to above stated warranty shall be limited to Seller, at Seller's sole discretion, either repairing the product, replacing the product with a like or similar product, or refunding the purchase price of the nonconforming product, provided that written notice of said nonconformance is received by Seller within the time periods set forth below:

a. For all software products, including licensed programs, ninety (90) days from date of initial delivery to Buyer;

b. For all hardware products (excluding batteries), including complete systems, fifteen (15) months from date of initial delivery to Buyer, subject to the additional conditions of paragraph c) below;

c. In the event that Buyer's returned product is a Discontinued product and is not repairable for any reason, Seller may elect to replace it with like or similar product that is, in Seller's sole judgment, the closest equivalent to the returned product. Seller does not warrant that such replacement product will be an exact functional replacement of the returned product.

Further, all products warranted hereunder for which Seller has received timely notice of nonconformance must be returned FOB Seller's plant no later than thirty (30) days after the expiration of the warranty periods set forth above.

These warranties provided herein shall not apply to any products which Seller determines have been subjected, by Buyer or others, to operating and/or environmental conditions in excess of the limits established in Seller's published specifications or otherwise have been the subject of mishandling, misuse, neglect, improper testing, repair, alteration or damage. THESE WARRANTIES EXTEND TO BUYER ONLY AND NOT TO BUYER'S CUSTOMERS OR USERS OF BUYER'S PRODUCT AND ARE IN LIEU OF ALL OTHER WARRANTIES WHETHER EXPRESS, IMPLIED OR STATUTORY INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL SELLER BE LIABLE FOR INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES. Seller's liability for any claim of any kind shall in no case exceed the obligation or liability specified in this Warranty clause.

Technical Assistance and Service
Seller's warranty as herein set forth shall not be enlarged, diminished or affected by, and no obligation or liability shall arise or grow out of, Seller's rendering of technical advice, facilities or service in connection with Buyer's order of the goods furnished hereunder. Products returned for warranty service, but which are found to be fully functional and in conformance with specifications may be subject to a nominal service charge and return freight charges. Periodic recalibration of products, if required, is the responsibility of Buyer and is not provided under this Warranty.

Online Support
Online support is available through technical support links on Intelligent Instrumentation’s 24-hour World Wide Web site at http://www.lanpoint.com. The site contains information on Intelligent Instrumentation’s products, new developments, announcements, application notes, application examples, and other useful information. The site and support areas continue to grow as new products, updates, and features are added.

Email Support
Intelligent Instrumentation’s technical support can be reached via email. When sending an email message, be sure to include complete contact information, the product model/part number with third-party accessory information, and a detailed description of the problem, to the support group: support@lanpoint.com
Static Sensitivity
Seller ships all static-susceptible products in anti-static packages. Seller’s Warranty as herein set forth shall not cover warranty repair or replacement for products damaged by static due to Buyer’s failure to use proper protective procedures when handling, storing, or installing products.

Trademarks
LANpoint Computers®, LANpoint®, Intelligent Instrumentation® are trademarks of Intelligent Instrumentation, Inc. Other products or brand names are trademarks or registered trademarks of their respective companies.

Use of Equipment
Intelligent Instrumentation Inc. assumes no responsibility for any direct, indirect or consequential loss or damages resulting from misuse of the equipment or for improper or inadequate maintenance of the equipment or for any such damage or loss resulting from the use of other equipment, attachments, accessories, and repairs at any time made to or placed upon the equipment or any replacement thereof. Furthermore, Intelligent Instrumentation Inc. makes no representations or warranties, either expressed or implied, in connection with the use of the equipment in the event it is improperly used, repaired or maintained.

FCC Radio Frequency Interference Statement
This equipment generates and uses radio frequency energy, and may cause interference to radio or television reception.

Per FCC rules, Part 15, Subpart J, operation of this equipment is subject to the conditions that no harmful interference is caused and that interference must be accepted that may be caused by other incidental or restricted radiation devices, industrial, scientific or medical equipment, or from any authorized radio user.

The operator of a computing device may be required to stop operating his device upon a finding that the device is causing harmful interference and it is in the public interest to stop operation until the interference problem has been corrected.

The user of this equipment is responsible for any interference to radio or television reception caused by the equipment. It is the responsibility of the user to correct such interference.

European CE Certification
European CE certification is as described on the Declaration of Conformity that ships with each LANpoint PLUS terminal.

Revision History for the LANpoint PLUS Developer’s Manual

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>020105</td>
<td>Release</td>
</tr>
<tr>
<td>1.1</td>
<td>032306</td>
<td>Cover change; include LANpoint TIME; remove outdated information</td>
</tr>
<tr>
<td>1.2</td>
<td>073107</td>
<td>RF Ethernet and minor corrections and changes</td>
</tr>
<tr>
<td>1.3</td>
<td>071508</td>
<td>Added Windows CE 5.0</td>
</tr>
</tbody>
</table>
Table of Contents

Chapter 1  LANpoint PLUS: Setup & System Configuration
  1.1 About this Manual .................................................. 1-1
  1.2 LANpoint PLUS Hardware Features ............................... 1-2
    1.2.1 Circuit Boards ............................................. 1-2
    1.2.1.1 CPU Board ............................................. 1-2
    1.2.1.2 Internal Option Board ................................ 1-2
    1.2.2 Ethernet ................................................... 1-3
    1.2.3 Serial Communications Ports ............................... 1-3
    1.2.4 Bar Code/AutoID input ................................... 1-3
    1.2.5 Digital Input/Output ...................................... 1-4
    1.2.6 Keyboard IMPORTANT NOTE ................................ 1-4
    1.2.7 Display .................................................... 1-5
  1.3 LANpoint PLUS Operation and Configuration .................... 1-5
    1.3.1 Windows CE Operating System ............................... 1-5
    1.3.2 Initial Power-up Sequence ................................ 1-5
    1.3.3 Configuring IP Address Properties ......................... 1-6
      1.3.3.1 LANpoint PLUS Setup Utility ......................... 1-7
      1.3.3.2 Remote Display Utility ............................... 1-7
      1.3.3.3 Other Methods to Configure the TCP/IP Properties 1-10
    1.3.4 File Transfer Methods ..................................... 1-11
    1.3.5 Accessing Shared Directories .............................. 1-12
    1.3.6 Instructions to Create a Shortcut in Windows CE .......... 1-13
    1.3.7 Saving the Contents of the Windows CE Registry .......... 1-13
      1.3.7.1 Using the RegFlushKey Function Call ............... 1-14
      1.3.7.2 Factory Default Registry File ....................... 1-14
    1.3.8 Permanent Storage Considerations ........................ 1-14

Chapter 2  Developing Custom Programs
  2.1 Selecting the Correct Development Tools for your Environment ... 2-1
    2.1.1 eMbedded Visual Tools V4.0 ................................ 2-2
    2.1.3 Visual Studio 2008 ......................................... 2-3
  2.2 Development Overview ............................................ 2-3
    2.2.1 Application & Data Transfer Methods ...................... 2-3
  2.3 Development with Microsoft eMbedded Visual C++ v4.0 .......... 2-4
    2.3.1 Terminal Setup - eMbedded Visual Tools v4.0 ............ 2-4
    2.3.2 Installing eMbedded Visual Tools V4.0 .................. 2-4
    2.3.3 Installing the Intelligent Instrumentation SDK .......... 2-5
    2.3.4 Programming Example - eMbedded Visual C++ v4.0 .......... 2-7
2.4 Development with Microsoft Visual Studio.NET 2003, Visual Studio 2005 or Visual Studio 2008 ........................................... 2-15
  2.4.1 Installing Microsoft Visual Studio. ......................... 2-15
  2.4.2 Installing the LANpoint PLUS SDK ....................... 2-15
  2.4.3 Sample Microsoft Visual Studio.NET 2003 development session. 2-15
2.5 ActiveSync Reference ........................................ 2-23
  2.5.1 Terminal Setup - ActiveSync ............................... 2-24
  2.5.2 Sample Microsoft Visual Studio.NET 2003 Application download via ActiveSync ........................................ 2-29
2.6 Uninstalling the SDK ......................................... 2-32
2.7 MSMQ and SQL CE support .................................... 2-32
  2.7.1 Microsoft Message Queue (MSMQ) ......................... 2-32
    2.7.1.1 MSMQ Configuration ................................. 2-33
    2.7.1.2 Minimum Server Configuration ..................... 2-34
    2.7.1.3 Sample Code ....................................... 2-34
  2.7.2 SQL CE .................................................. 2-34
    2.7.2.1 Using SQL CE - Local Database .................... 2-35
    2.7.2.2 Using SQL CE - Remote Database Via XML ........ 2-36
    2.7.2.3 Using SQL CE - Database Replication ............ 2-36
    2.7.2.4 Using SQL CE - Remote Data Access ............. 2-37
    2.7.2.5 Additional SQL CE Reference Material: ........... 2-37

Chapter 3 Utility Programs
  3.1 Setup Utility ............................................... 3-1
  3.2 Remote Display Utility ..................................... 3-4
  3.3 CEPad Text Editor Utility .................................. 3-5
  3.4 Wedge Utility ............................................... 3-5
    3.4.1 Configuring WedgeCE Parameters ....................... 3-6
    3.4.2 WedgeCE Default Parameters ......................... 3-7
  3.5 Programmable Function Key Utility ........................ 3-7
    3.5.1 Creating a New Programmable Function Key File ..... 3-8
    3.5.2 Editing a Programmable Function Key File .......... 3-9
    3.5.3 Testing a Programmed Function Key .................. 3-9
    3.5.4 Using a PFK .......................................... 3-10
    3.5.5 Deleting a Function Key ............................... 3-10
  3.6 DebugLauncher Utility ....................................... 3-10
  3.6.1 How it Works ........................................... 3-10
  3.7 Time Synchronization ....................................... 3-11
    3.7.1 Time Synchronization for Windows CE.NET 4.2 ...... 3-11
      3.7.1.1 Executing the Time Synchronization Program .... 3-11
      3.7.1.2 Handling Synchronization Failures .............. 3-12
3.7.1.3 Command Line Arguments ................................................. 3-12
3.7.2 Time Synchronization for Windows CE 5.0 ................................. 3-14
3.7.2.1 Time Synchronization Configuration Program .......................... 3-14
3.8 Monitor Program ............................................................... 3-16
3.8.1 Required Tools .............................................................. 3-16
3.8.2 Configuring the Communications Program ................................. 3-17
3.8.3 Starting a MonitorCE Program Session .................................... 3-18
3.8.4 MonitorCE Program Commands ........................................... 3-18
3.9 Remote Manager Utility ....................................................... 3-20
3.9.1 Enabling WebDevice for Remote Management .............................. 3-21
3.9.2 Using the Remote Manager ................................................ 3-22
3.9.2.1 Management ............................................................ 3-23
3.9.2.2 Security Manager: Allowing Access to Folders .......................... 3-24
3.10 SerialSocket Utility ............................................................ 3-26
3.10.1 Port Configuration for SerialSocket Utility ............................... 3-26
3.10.2 Viewing the Status of the SerialSocket Utility ............................ 3-26
3.10.3 Testing SerialSocket with HyperTerminal .................................. 3-27
3.11 SNMP .......................................................................... 3-27
3.12 LCmdSet Utility ............................................................... 3-28
3.12.1 Running LCmdSet ............................................................ 3-28
3.12.1.1 Host Computer: Launching a Telnet Client ............................ 3-28
3.12.2 Issuing Commands .......................................................... 3-28
3.12.2.1 Examples .............................................................. 3-28
3.12.3 Set Commands ............................................................... 3-29
3.12.4 Show Commands ............................................................. 3-29
3.12.5 Change Commands .......................................................... 3-32
3.12.6 Services Commands ......................................................... 3-34
3.12.7 Operational Commands .................................................... 3-35
3.13 Thin Client: Terminal Emulation ............................................. 3-35
3.13.1 Text-based Terminal Emulation ............................................ 3-36
3.13.1.1 Text-Based Terminal Emulation Installation Instructions ................ 3-36
3.13.1.2 Text-Based Terminal Emulation Un-Install Instructions ............... 3-37

Chapter 4 LANpoint PLUS APIs
4.1 Overview ................................................................. 4-1
4.1.1 API Header Files and Import Libraries .................................... 4-2
4.2 Digital I/O API .............................................................. 4-2
4.2.1 Input and Output Channels ............................................... 4-2
4.2.1.1 Input Channels ....................................................... 4-2
4.2.1.2 Output Channels ..................................................... 4-3
4.2.2 Required Files ................................................. 4-3
4.2.3 Return Values ................................................ 4-3
4.2.3.1 When Errors Occur ....................................... 4-3
4.2.4 Digital I/O Input Functions .............................. 4-3
4.2.4.1 Return the State of a Single Channel ............... 4-4
4.2.4.2 Return the Current State of All Input Channels .... 4-5
4.2.4.3 Configure the Mode of an Input Channel .......... 4-6
4.2.4.4 Obtain the Current Configuration Mode of a Channel 4-7
4.2.4.5 Store Input Modes and Output States ................ 4-7
4.2.4.6 Retrieve Input Modes and Output States ............. 4-8
4.2.4.7 Return Data from a Counter Bit ................. 4-9
4.2.4.8 Input Bit Transitions: Trigger Activation ......... 4-9
4.2.5 Digital I/O Output Functions ......................... 4-10
4.2.5.1 Control a Single Output Channel .............. 4-10
4.2.5.2 Turn All Outputs on or off .................... 4-12
4.2.6 Digital I/O Error Codes ............................. 4-12

4.3 Time Synchronization API ........................................ 4-14
4.3.1 Time Server Functions ....................................... 4-15
4.3.1.1 Network Name or IP Address of the Time Server .... 4-15
4.3.1.2 Set Protocol to use with the Time Server .......... 4-16
4.3.1.3 Set Timeout for a UDP Protocol .................... 4-16
4.3.1.4 Get Server Time ..................................... 4-16
4.3.1.5 Return Current LANpoint PLUS System Time .... 4-17
4.3.1.6 Calculate Time Difference ........................... 4-17
4.3.1.7 Synchronize the System Clock with the Server Clock 4-17
4.3.2 Time Sync API Error Codes in eMbedded Visual C++ 4-18

4.4 Display API ...................................................... 4-18
4.4.1 Display API Functions ................................. 4-19
4.4.1.1 Position the cursor - cursorxy ........................ 4-20
4.4.1.2 Get the current position the cursor - get_cursorxy .. 4-21
4.4.1.3 Show or hide the underline cursor .................. 4-21
4.4.1.4 Return the current state of the underline cursor .... 4-22
4.4.1.5 Show or hide the block cursor .................... 4-22
4.4.1.6 Return the current state of the block cursor ....... 4-23
4.4.1.7 Display text on the LCD display .................... 4-23
4.4.1.8 Display text on the LCD display, with optional character mapping 4-24
4.4.1.9 Display a string at current cursor position ........ 4-25
4.4.1.10 Clear the LCD display .............................. 4-26
4.4.1.11 Clear a line on the LCD .............................. 4-26
4.4.1.12 Clear a portion of a line on the LCD ............. 4-27
4.4.1.13 Clear a line on the LCD, from cursor position to end-of-line 4-28
4.4.1.14 Enable LCD display .................................. 4-28
4.4.1.15  Turn LCD backlight on ......................... 4-29
4.4.1.16  Set LCD backlight to automatic control by OS ............ 4-29
4.4.1.17  Define custom LCD character display shape bitmaps ....... 4-30
4.4.1.18  Unicode character set remapping ..................... 4-30

4.5  Display Subclass API ........................................... 4-31
4.5.1  Display Subclass API Functions ............................. 4-32
   4.5.1.1  Associate an EDIT control with the LCD display ....... 4-32
   4.5.1.2  Associate a checkbox BUTTON control with the LCD display .... 4-34

4.6  Reboot API ..................................................... 4-35
   4.6.1  Reboot Calls ............................................. 4-35
      4.6.1.1  Reboot Function ................................. 4-35

Index
Chapter 1 LANpoint PLUS: Setup & System Configuration

This manual provides information on how to program the LANpoint PLUS terminal and is written for developers having knowledge of the Microsoft Windows CE operating system and Microsoft eMbedded Visual Tools or Microsoft Visual Studio [.NET]. [.NET] was part of the product name in early versions; it’s still usable and instructions are included herein.

In addition to this manual, there is a LANpoint PLUS Quick Start Guide, and a LANpoint PLUS Installation & Maintenance Manual. There are also manuals for specific mounting accessories, Vehicle-Mount (VM) models, and the LANpoint TIME which are shipped with those products or available on the Web.

All manuals are available as Adobe Acrobat files located on Intelligent Instrumentation’s web site at http://www.lanpoint.com on the main overview page for the specific LANpoint PLUS model (stationary, vehicle-mount, LANpoint TIME) being used. Sample programs are also provided on this web site.

1.1 About this Manual

The LANpoint PLUS Developer's Manual contains the following information:

Chapter 1 LANpoint PLUS: Setup & System Configuration
This chapter provides a brief overview of the terminal’s hardware and software features including: operation of the terminal from the initial power-up sequence, configuring the terminal including the remote display and manipulation of the Windows CE GUI interface via that remote display, transferring files, creating shared directories, creating shortcuts, and saving configuration settings.

Chapter 2 Developing Custom Programs
This chapter contains installation information for the Microsoft eMbedded Visual Tools 4.0 software, Microsoft Visual Studio (.NET), and the LANpoint PLUS SDK (Software Development Kit). It includes information on writing and compiling custom programs. It provides instructions on launching eMbedded Visual Tools, Microsoft Visual Studio (.NET), configuring the platform manager, and file transfer methods.

Chapter 3 Utility Programs
This chapter explains the utility programs: Setup utility, remote display, CEPad text editor, keyboard wedge, programmable function keys, DebugLaucher, time synchronization, Monitor program, Remote Manager, SerialSocket, and LCmdSet. The LANpoint PLUS extensions to standard SNMP are also documented, as are several terminal emulation applications.

Chapter 4 LANpoint PLUS APIs
This chapter contains information about the following APIs: digital I/O, time synchronization, text display and subclassing, and reboot.
1.2 LANpoint PLUS Hardware Features

1.2.1 Circuit Boards
The standard LANpoint PLUS terminal contains a CPU board with Digital I/O functionality. An Internal Option Board with a PCMCIA slot (with or without factory-installed RF WiFi radio) and a CompactFlash slot is available. This section includes a brief description of these boards. For more information about hardware, see the LANpoint PLUS Installation and Maintenance Manual.

1.2.1.1 CPU Board
The CPU board includes an embedded microcontroller, Flash and RAM memory, and supporting hardware for the display, keypad, external connectors, and Internal Option Board.

1.2.1.1.1 Embedded Microcontroller
The CPU board features the Intel XScale PXA255 200MHz embedded microcontroller. The Windows CE.NET 4.2 or Windows CE 5.0 operating system and boot code run on this CPU and oversee all of the terminal’s functions.

1.2.1.1.2 Memory: RAM
The terminal contains 32MB of SDRAM

1.2.1.1.3 Memory: FLASH, Storage Card Slots
FLASH memory provides 32MB of non-volatile memory for the operating system, utilities, application program(s), and application data storage. The terminal ships from the factory with approximately 8 MB free space for application programs and local application data storage, which can be expanded via a Compact Flash card and/or PCMCIA FLASH card in models that include the Internal Option Board. Standard FLASH memory can also be freed-up by deleting unused utilities and other files not used in application deployment.

Special folders/subdirectories in FLASH allow applications to be executed on start-up and allow files to be copied to other storage sections of the RAM filesystem, such as the Windows desktop or the Start menu.

Windows CE 5.0 models have the whole flash filesystem mounted as the root of the filesystem; as such, the \windows folder and every other folder is stored in persistent flash memory. This is unlike the Windows CE.NET 4.2 models, where everything except the \flash folder is in the RAM filesystem and therefore only persists until it is rebooted.

1.2.1.2 Internal Option Board
A factory-installed Internal Option Board is available on all LANpoint PLUS models. The Internal Option Board has one Compact Flash slot and one PCMCIA slot which supports PC
Card devices including: ATA, WiFi RF Ethernet, and Compact Flash (with an adapter card), and PCMCIA Flash. Both 3.3V and 5V PC cards are supported.

The Compact Flash slot supports one Compact Flash memory card of any size for additional non-volatile data storage. The Compact Flash slot is accessible to the developer by removing the rear cover of the terminal; cards can be hot-swapped.

The PCMCIA slot supports one PCMCIA card. It is primarily used for RF Ethernet (IEEE802.11b/g, WiFi), though it also supports FLASH memory cards.

The PCMCIA slot is accessible by removing an access panel on the back cover of the terminal.

See the *LANpoint PLUS Installation and Maintenance Manual* or the *LANpoint TIME Installation and Maintenance Manual* for more information about the location and other features of the slots on the Internal Option Board.

**1.2.2 Ethernet**

Every terminal has an RJ45 10/100 Base-T Ethernet port controlled by a SMC91C111 Ethernet controller.

In addition, a PCMCIA slot on the Internal Option Board on WiFi models is populated with the IEEE802.11b/g wireless radio frequency (RF) Ethernet card option. See the *LANpoint PLUS Installation and Maintenance Manual* or the *Installation Manual for Vehicle-Mount LANpoint PLUS* for configuration, and additional information.

**1.2.3 Serial Communications Ports**

COM1, COM2 and COM3 are used for normal serial communications. The COM port factory default communications parameters are 9600 baud, no parity, 8 data bits, 1 stop bit, and no flow control. An easy way to show the current settings or change the settings is to use the LCmdSet Telnet server utility as described in 3.12.1, *Running LCmdSet*.

AutoID devices like laser scanners, can be connected to these ports as well as the keyboard port- see more information immediately below. The WedgeCE program can be used for any or all of the COM ports to wedge input data into the keyboard datastream.

The COM1 serial port can be used to configure the unit’s network settings and real-time clock using the MonitorCE utility. See section 3.8, *Monitor Program*.

**1.2.4 Bar Code/AutoID input**

The terminal supports autoID readers from many manufacturers.

- Standard "keyboard wedge" devices (magnetic stripe and barcode badge slot readers, wands, lasers, imagers, etc.) via the PS/2 keyboard port. Multiple devices and external keyboard(s) can be used with 'Y' cables. Power is supplied through the PS/2 port. The autoID data is placed directly in the keyboard datastream.
• Serially connected devices such as 2D scanners, slot readers, biometric readers, proximity readers and the like via COM1, COM2, and/or COM3.
• A "wedge" software utility (wedgece.exe) included with each terminal may be used to "wedge" serial input data from any or all COM ports into the keyboard datastream.

To power an autoID device (or any accessory, for that matter), each LANpoint PLUS terminal has a connector with 4 pins that can be used to provide ground, 3.3VDC, 5VDC, and/or 12VDC via a jumper wire. Alternatively, pin 9 of any of the three COM ports can be configured to provide +5VDC, common on many serial scanners, via a jumper. See the LANpoint PLUS Installation and Maintenance Manual for more information.

1.2.5 Digital Input/Output

Every LANpoint PLUS terminal provides eight digital input channels and eight digital output channels. For ease of field wiring, a digital I/O termination panel accessory is available. For more information, see the LANpoint PLUS Installation and Maintenance Manual or the LANpoint TIME Installation and Maintenance Manual.

1.2.6 Keyboard IMPORTANT NOTE

Throughout this manual there will be references concerning use of the keyboard for setup, configuration, and maintenance activities. Since many characters cannot be typed on the special reduced keypad, an external PS/2 keyboard must be used temporarily for these purposes.

IMPORTANT: On reduced numeric/function keypad models, temporarily connect an external keyboard to the terminal for configuration and installation activities. Use any keyboard with a PS/2 style connector.

The terminal comes standard with a 51-key sealed elastomeric QWERTY keyboard or a reduced-key function and numeric keypad (standard on the LANpoint TIME model) integrated into the case. The keyboard is a long-life industrial design, rated for 5 million key presses.

Eight function keys are mappable to 32 separate functions using modifier keys (alt, ctrl, shift). The function keys (and almost any key) can be programmed to insert a string of characters into the keyboard buffer when pressed, or to launch a program. See section 3.5, Programmable Function Key Utility of this manual for more information.

To use an external keyboard and/or a keyboard wedge autoID reader, connect the cable before applying power to the terminal. Windows CE does not support hot swap. Connect the keyboard cable to the keyboard mini-DIN6 connector, J5, and restart the LANpoint PLUS terminal. If multiple keyboard/keypad wedge devices are desired, use the 'Y' cable that is usually supplied with that device or easily procured at electronic stores.
1.2.7 Display

The LANpoint PLUS terminal has a 2-line by 40 character backlit LCD display. This integral display allows the terminal to be very small and very functional.

Developers (and in some applications, installers) will want to become familiar with the Remote Display Utility as described in Section 3.2, Remote Display Utility.

By default, the backlight is always on. To turn it off, or to allow it to turn off during periods of inactivity, use the Display applet in the Control Panel via the Remote Display Utility (see Section 3.2, Remote Display Utility). For instance, if the battery backup accessory is installed, you may wish to select the Backlight tab, select the Automatically turn off backlight while on battery power checkbox, select the desired interval, and click the OK button.

1.3 LANpoint PLUS Operation and Configuration

The topics in this section include:

- the Microsoft Windows CE.NET 4.2 and Windows CE 5.0 operating systems
- the initial power-up sequence
- using the Windows CE GUI interface via the remote display utility
- configuring the terminal
- transferring files from the development computer to the terminal
- creating shared directories to upload and download files to and from the terminal
- creating a shortcut with or without command line parameter(s)
- saving the contents in the Windows CE registry
- creating folders on the terminal for permanent storage

1.3.1 Windows CE Operating System

The LANpoint PLUS runs the Microsoft Windows CE.NET version 4.2 or the Windows CE version 5.0 operating system. The operating system configuration includes support for the hardware devices covered in Section 1.2, LANpoint PLUS Hardware Features, Compact Framework (.NET Compact Framework version 1.0 with CE.NET 4.2 or .NET Compact Framework version 3.5 with CE 5.0), and other options and common accessories.

1.3.2 Initial Power-up Sequence

The fixed-mount terminal does not have a power switch. The terminal receives power from an external power source via an adapter when 12VDC power is connected to the Power Input connector on the rear of the terminal.

Vehicle-mount versions have a power switch on the rear of the vehicle-mount chassis.

The LCD backlight will come on when the terminal is powered-up.
During the initial load, Windows CE establishes interrupt handlers and attempts to load the registry from non-volatile storage if the internal jumper JP13 is installed. If JP13 is not installed, the terminal uses the factory default registry.

**Note:** On Windows CE.NET 4.2 models, after the OS initializes, it copies the contents of any directory structure in `\Flash\My Computer` from the FLASH memory to the Windows CE.NET root directory in RAM. This can be used, for instance, to put a program on the Desktop by copying the program to `\Flash\My Computer\Windows\Desktop\progname.exe`. Any programs in `\Flash\Startup\` are launched.

Windows CE 5.0 models have the whole flash filesystem mounted as the root of the filesystem; as such, the `\windows` folder and every other folder is stored in persistent flash memory. This is unlike the Windows CE.NET 4.2 models, where everything except the `\flash` folder is in the RAM filesystem and therefore only persists until it is rebooted. As a result, you may use the `\Windows` folder and all its subfolders (instead of `\Flash\My Computer\Windows\Desktop\progname.exe`, for instance) and the `\Startup` folder (instead of `\Flash\Startup\`).

**WARNING:** Be careful not to duplicate startup programs, as both methods are built into CE 5.0 units.

The system boots up and performs hardware verification. A checkerboard pattern will be displayed briefly, and the speaker emits the startup tone. A screen showing the OS version number and the IP address, similar to that shown below, will be displayed.

![LANpoint PLUS Successful power-up](image)

During power-up, the terminal launches the `MonitorCE` utility which allows certain configuration settings via commands sent to serial port `COM1`. A description of `MonitorCE` can be found in Section 3.8, *Monitor Program*.

### 1.3.3 Configuring IP Address Properties

The LANpoint PLUS default networking configuration is wired Ethernet using DHCP. The DHCP assigned address (or factory default address, if there is no DHCP server available) is displayed on the initial "Version" screen.

There are several ways to initially access the LANpoint PLUS from a Windows PC in order to change the networking default and other basic configurable features. The most direct method is described here. It entails the use of the LANpoint PLUS Setup Utility (see section 3.1, *Setup*
Utility) and also includes some basic steps using the Remote Display Utility (see Section 3.2, Remote Display Utility).

Some configuration tasks, including the setting of RF Ethernet properties can only be accomplished using the Remote Display Utility.

Described below is the most direct configuration of the terminal, using the

- Setup Utility to initially set wired Ethernet properties and launch the
- Remote Display Utility for full access to the LANpoint PLUS’ Windows CE Graphical User Interface

1.3.3.1 LANpoint PLUS Setup Utility

1. To start the Setup Utility on the LANpoint PLUS, press F16 (press the 2nd key and release it, then press the F8 key). The LANpoint PLUS LCD will briefly show:

   ![LANpoint PLUS Setup Utility Startup Screen](image)

   **LANpoint PLUS Setup Utility Startup Screen**

   2. followed by a prompt for a command:

   ![LANpoint PLUS Setup Utility Command Screen](image)

   **LANpoint PLUS Setup Utility Command Screen**

   3. If DHCP is used to automatically set the IP address, skip to step 6. To set a hardcoded IP address type nixxxx.xxx.xxx.xxx at the Setup> prompt, where x = IP address. A full list of Setup commands can be found in Section 3.1, Setup Utility.

   4. At the Setup> prompt type nsxxxx.xxx.xxx.xxx where x = subnet mask.

   5. At the Setup> prompt type ngxxxx.xxx.xxx.xxx where x = gateway address. Reboot the terminal for the changes to take effect.

   6. For wired Ethernet, the configuration of the IP address is complete. Otherwise, if you are using RF wireless Ethernet, make sure the RF card is installed in the PCMCIA slot under the rear cover. This can be done with the terminal powered or unpowered.

1.3.3.2 Remote Display Utility

7. Configure the wireless parameters using the Remote Display Utility. First, copy and run the program CERHOST.EXE on a development PC connected to the network.
LANpoint PLUS: Setup & System Configuration

*CERHOST.EXE* can be found on the SDK CD, or downloaded from the www.lanpoint.com overview page for the specific terminal model (stationary, vehicle-mount, LANpoint TIME) model in use.

8. If the terminal has not previously been configured to use RF Ethernet, make sure the wired Ethernet cable is still attached to the LANpoint PLUS. Repeat steps 1 and 2. At the Setup> prompt, type `rdhostxxx.xxx.xxx.xxx` where x = the IP address of the development PC.

9. At the Setup> prompt, type `rdstart`. The LANpoint PLUS Graphical User Interface will appear in the *CERHOST* window on the development PC.

10. In the development PC’s *CERHOST* window, go to `Start/Settings/Network and Dial-up connections`. In the pop-up dialog, double-click on the SDCCF10G1 icon and configure the IP address and Name server for the wireless connection.

WLAN IP address Configuration

11. In the development PC’s *CERHOST* window, run the *Summit Client Util.* dialog to set configuration parameters like ESSID, WEP, and security parameters. Double-click on the RF Ethernet network icon in the system tray to open the dialog shown below, or

---

CERHOST window on development PC - WLAN client configuration utility
select Start/Programs/Summit/SCU to do the same. The version that runs on a
Windows CE 5.0 unit differs slightly - for instance, the Config tab is titled Profile, and the
information and buttons are re-arranged, but the same functionality is provided.

12. The Summit Client Utility dialog opens in the CERHOST window, as shown below.

![Summit Client Utility dialog](image1.png)

**Note:** The default password to modify the parameters with this dialog is
SUMMIT, all capitalized. You can set it to your own value in the Global Settings
dialog.

This utility shows and, when logged-in as Administrator, allows configuration of settings
relating to the WiFi connectivity. Press the Admin Login button to enter the password to
make changes. The factory-default password is SUMMIT, in all capital letters.

13. Once the password is entered, all functions supported by the Summit Client Utility are
accessible and configurable in the CERHOST window. For instance, the figure above shows
the Config (Profile) dialog where new configurations can be added, and the SSID, client
name, and other configuration parameters can be set. The Commit button must be pressed
to record any configuration changes.
Currently supported EAP types are: LEAP, EAP Fast, PEAP-MSCHAP, and PEAP-GTC. Currently supported Encryption methods are: Manual WEP, Auto WEP, WPA PSK, WPA TKIP, WPA2 PSK, WPA2 AES, CCKM TKIP, CKIP Manual, and CKIP Auto.

14. Another useful tool is the Diagnostic dialog as shown below. Other nodes on the network can be PINGed, the terminal can be re-connected to the network, and the IP address can be viewed/released/renewed; all useful tools for wireless network troubleshooting. Site Survey is not supported in the CE 4.2 release of the SCU, and does not exist on the CE5.0 release.

15. The other tabs, like Status, show client name, signal strength, radio channel, communications speed, MAC address, and others. Global settings shows and allows the setting of global parameters like roaming parameters, ping parameters, and the Admin password.

16. After the wireless parameters are set, run /FLASH/CEFlush.exe to save the registry in persistent memory (see 1.3.7, Saving the Contents of the Windows CE Registry). On Windows CE 5.0 models, Restart (Press the Start button icon) will warm-boot the terminal, after saving the registry.

17. Reboot to test the wireless Ethernet connection.

1.3.3.3 Other Methods to Configure the TCP/IP Properties

Other ways to change the wired IP address, subnet mask, and gateway, to set DHCP server parameters, DNS parameters, WINS servers, and those types of IP addresses include:

---

**Note:** On Windows CE 4.2 models, CEFflush must be run to save the changes in the registry in non-volatile storage. Failure to do so will result in loss of settings when the terminal is re-booted.

On Windows CE5.0 models, either a warm boot (restart) or CEFflush can be used to save the changes.

---

1-10
1. The Control Panel in Windows CE, in conjunction with the Remote Display Utility as specified above in Section 1.3.3.2, Remote Display Utility

**Note:** On Windows CE 4.2 models using the Control Panel method, CEFlush must be run to save the changes in the registry in non-volatile storage. Failure to do so will result in loss of settings when terminal is re-booted.

On Windows CE5.0 models, either a warm boot (Restart) or CEFlush can be used to save the changes.

2. The Monitor Utility program. See MonitorCE instructions in section 3.8, Monitor Program

3. The embedded Telnet Server utility. See LCmdSet instructions in 3.12, LCmdSet Utility. The LCmdSet Utility is in the Startup folder when the terminal ships from the factory, so it does not have to be manually launched.

### 1.3.4 File Transfer Methods

To transfer files from a development computer or any computer to the LANpoint PLUS terminal, the following methods can be used.

- Use the Remote File Viewer in Microsoft eMbedded Visual Tools 4.0 or Microsoft Visual Studio [.NET] to transfer files as shown in the embedded Visual Tools V4.0 example in Section 2.3.4, Programming Example - eMbedded Visual C++ v4.0.
- Use Build/Deploy Solution option in Microsoft Visual Studio.NET 2003 and later.
- Use the Remote Manager to upload the application using a Web interface. (see section 3.9, Remote Manager Utility)
- Define shared network folders (see section 1.3.5, Accessing Shared Directories).
- Use Microsoft ActiveSync (see Section 2.5, ActiveSync Reference) to transfer the desired files.

For occasional or one-time file transfer, the PCMCIA (PC card) or Compact Flash slot can be utilized. These slots allow hot-swapping of the cards. The slot(s) need to be accessible, so covers and/or mountings may have to be removed or partially disassembled.

- A Compact Flash card can be plugged directly into the Compact Flash slot of the terminal to use as a removable media to transfer files between the terminal and another computer.
- A FLASH PC card can be plugged directly into the PCMCIA (PC card) slot.
- A Compact Flash card can be plugged into a PC card/Compact Flash adapter that is then plugged into the PCMCIA slot.
- The card shows-up as a folder named \Storage Card in Windows Explorer.
- If both slots have cards installed, the second slot shows-up in Windows Explorer as a folder called \Storage Card2.
1.3.5 Accessing Shared Directories

Accessing a shared directory on a PC on the Ethernet network allows you to copy files between the terminal and a PC, as well as to update files in the shared directory on the PC, as if that shared directory were stored locally on the terminal.

Note: On the LANpoint PLUS and on a development PC, execute the Remote Display Utility as described in Section 1.3.3.2, Remote Display Utility. Use the utility to perform the following steps on the LANpoint PLUS.

1. It is a good idea to assign each LANpoint PLUS terminal a unique name on your network. If this has already been done, skip to step 7.

2. Reference the Note above. Click on the Start button and select Control Panel.

3. Double-click the System icon to open the System Properties dialog.

4. Click on the Device Name tab. Enter a unique identifier in the Device name: field. Enter a description in the Device description: field if you wish.

5. Press the OK button in the System Properties dialog.

6. Run \Flash\CEFlush to save the device name settings to non-volatile FLASH memory. On Windows CE 5.0 models, Restart (Press the Start button icon) can also be used to both save the settings and warm boot the terminal.

7. In the Command Prompt window on the LANpoint PLUS terminal (Start -> Programs -> Command Prompt), enter the command:

   net use Test \ComputerName\SharedDirectoryName

   where ComputerName is the computer with which you want to share data. The ComputerName and SharedDirectoryName is defined on the PC beforehand; these values and the methods to set them (and look them up) are operating system dependent. See your system administrator if either or both are unknown to you.

8. A dialog opens, asking for a user name and password. Enter the user name and password for the host computer. Some computers may require an entry in the Domain field. Leave this field empty if your computer does not require an entry.

9. Click the OK button.

10. A folder named Test now exists on the terminal under the Network folder. Below that folder is displayed the contents of the host computer’s shared folder.
11. Use Windows CE.NET Explorer or any other program to copy, modify, delete, and so on, as if the files were all stored locally.

**Caution:** On Windows CE.NET 4.2 models, the content of the root directory is not saved when the LANpoint PLUS terminal is rebooted. To save files on the terminal, be sure to copy your files to non-volatile memory for permanent storage (anywhere under the \Flash folder).

On Windows CE 5.0 models, all folders are stored in non-volatile memory.

### 1.3.6 Instructions to Create a Shortcut in Windows CE

1. Reference the Note above. Select (highlight) the executable program using Windows Explorer or My Computer that is shown on the desktop.

2. Copy the file using the *Edit* menu *Copy* item.

3. Go to the directory location where you want to place the shortcut and select *Edit* and select the *Paste Shortcut* item.

**Note:** To have an application automatically activate at startup on a Windows CE.NET 4.2 LANpoint PLUS terminal, you can create a shortcut to the application in the \Flash\Startup folder. On a Windows CE.NET 5.0 model, you may simply place it in the \Windows\Startup as you would with a desktop PC.

**WARNING:** Be careful not to duplicate startup programs, as both methods are built into CE 5.0 units.

4. To add parameters to the end of the shortcut command line, right click on the *shortcut* and select *Properties*.

5. Select the *Shortcut* tab. The *Target* field contains the command line for the program file which includes the path to the directory location and the name of the program.

6. Type the *parameters* at the end of the shortcut command line and click the *OK* button.

### 1.3.7 Saving the Contents of the Windows CE Registry

Changes that you make that affect the registry (in dialogs, during some program installations, changes of TCP/IP addresses, and so forth) are held in the Windows CE registry that is in *RAM*
As such, those changes are temporary and will not be in effect when the terminal is rebooted, unlike desktop Windows operating systems.

To make this kind of change permanent, the registry must be stored in (flushed to) non-volatile FLASH memory from which it can be loaded during startup.

- **CEFlush** is the program that is provided to write the registry to non-volatile memory. Find it in the \FLASH directory on the LANpoint PLUS. On Windows CE 5.0 models, **Restart** (Press the Start button icon) will warm-boot the terminal, after saving the registry.

```plaintext
TIP: There is no dialog or other indicator as to when CEFflush completes. If you are going to cold-boot - recycle power - allow 10 seconds after launch to be safe.

If you are launching from Windows Explorer, you can press the down arrow and when the display changes to the new cursor location, CEFflush has completed writing the registry file and you can safely cold-boot.

If you are going to warm-boot the terminal, you can do it at any time; CEFflush is synchronized with the restart function when the terminal is warm-booted.
```

### 1.3.7.1 Using the RegFlushKey Function Call

When writing a custom program, you can use the call `RegFlushKey()` to immediately flush the changes in the registry to non-volatile FLASH memory.

### 1.3.7.2 Factory Default Registry File

If the registry does not exist, or if jumper J13 is removed, the factory default registry settings are used.

```plaintext
NOTE: If you run CEFflush (or Restart) when the factory default registry is in effect, you will lose any registry settings and revert to the factory default at the next reboot, regardless of jumper J13.
```

### 1.3.8 Permanent Storage Considerations

All folders and storage areas on LANpoint PLUS terminals with Windows CE 5.0 are stored in non-volatile memory, as the root directory is in Flash memory. Everything will be persistent after a reboot.

The root directory on LANpoint PLUS terminals with Windows CE.NET 4.2, however, is RAM-based. Only the \Flash directory and directories under it are stored in non-volatile memory (and \Storage Card x directories, if Compact Flash cards are installed). **Be careful and keep in mind that the root directory is volatile storage and files copied there will not be**
available when the terminal reboots. This includes the \Windows folder under the root directory.

The LANpoint PLUS CE 4.2 terminal, however, has special routines built-in that can be used to make programs and other files available in the \Windows folder.

For example, to store a shortcut or executable program on the Windows CE.NET desktop:
1. Create a new folder in \Flash and name it My Computer (space is required between My and Computer).
2. In \Flash\My Computer create a new folder and name it Windows.
3. In \Flash\My Computer\Windows create a new folder and name it Desktop.
4. Paste the shortcut in Desktop and the icon will show on the desktop the next time the terminal is booted.
5. You can do this with any path you wish under the \My Computer folder.
Chapter 2  Developing Custom Programs

This chapter provides instructions for installing the development software and contains information on developing custom programs for the LANpoint PLUS terminal. Sample programs are provided on the Intelligent Instrumentation Web site www.lanpoint.com

Note: Throughout this chapter, when referring to steps performed on the LANpoint PLUS, use the Remote Display Utility where appropriate. Execute the Remote Display Utility as described in Section 1.3.3.2 Remote Display Utility.

2.1 Selecting the Correct Development Tools for your Environment

Developers have several options when making a decision regarding which development tools to use during Windows CE.NET v4.2 application development, eMbedded Visual Tools v4.0, Visual Studio.NET 2003, Visual Studio 2005.

For Windows CE 5.0 application development, the choices are eMbedded Visual Tools v4.0 and Microsoft Visual Studio 2005.

In general, the following table summarizes the tools that can be used with Windows CE.NET v4.2, which includes .NET Compact Framework (CF) v1.0 and those that can be used with Windows CE v5.0 which includes .NET Compact Framework v3.5.

<table>
<thead>
<tr>
<th>Target OS</th>
<th>Development Tool</th>
<th>Application Design</th>
<th>VS.NET 2003</th>
<th>VS 2005</th>
<th>VS 2008</th>
<th>eVTools 4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE.NET 4.2 w/CF calls</td>
<td>.NET CF 1.0</td>
<td>.NET CF 1.0</td>
<td>N</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CE.NET 4.2 w/o CF calls</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CE 5.0 w/CF calls</td>
<td>Y*</td>
<td>.NET CF 2.0*</td>
<td>.NET CF 3.5, 2.0</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CE 5.0 w/o CF calls</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Code will run, but doesn’t support debug

Legend: Y = code developed with this tool will run on terminal
N = code developed with this tool will NOT run on terminal or the tool cannot target this OS
.NET CF x.x = highest version of .NET CF call, if any, that can be included in the code
2.1.1  eMbedded Visual Tools V4.0

These tools are suitable for developing embedded Visual C++ applications only. The end product of the development process is a native ARM processor executable (.exe) which can be transferred to the unit and executed. All other things being equal, native code will typically outperform the managed code that is generated by Visual Studio.NET 2003, Visual Studio 2005 or Visual Studio 2008, but there are other trade-offs that can be considered:

**Advantages**

These tools are free from Microsoft.

Creates Native Code programs using the eMbedded Visual C++ programming language.

Can be used to develop applications for either Windows CE.NET 4.2 or Windows CE 5.0

**Disadvantages**

Cannot create programs using C# or Visual Basic.

The development tools do not provide for cabinet (.cab) file generation.

2.1.2  Visual Studio.NET 2003 & Visual Studio 2005

For Windows CE.NET 4.2 models, these tools are suitable for developing C# or Visual Basic applications. Visual Studio 2005 can also be used for native C++ development. The end product of the development process is an executable (.exe) which can be transferred to the unit and executed. They can also build a cabinet (.cab) file that can be transferred to the unit for easy multi-terminal deployments - users can double click on the .cab file and installation occurs as an automated process, similar as to what users see when they install new programs on a desktop PC.

**Advantages**

Can create distributable cab files suitable for installation.

Managed code is advantageous for deploying a program to different types of Windows CE.NET devices, as the managed code executable will run the same on different types of hardware, unless device-specific API’s are used.

User interface development is much easier using Visual Studio tools than is native code user interface development.

**Disadvantages**

Tools must be purchased from Microsoft.
2.1.3 Visual Studio 2008

These tools are suitable for developing C++, C# or Visual Basic applications for Windows CE 5.0 models. The end product of the development process is an executable (.exe) which can be transferred to the unit and executed. Visual Studio 2008 can also build a cabinet (.cab) file that can be transferred to the unit for easy multi-terminal deployments - users can double click on the .cab file and installation occurs as an automated process, similar as to what users see when they install new programs on a desktop PC.

Advantages and Disadvantages
Same as listed in section 2.1.2 above.

2.2 Development Overview

In general, the development process any of the development tool sets is the same:
• The developer installs the appropriate Microsoft development tools.
• The developer installs the Intelligent Instrumentation SDK that is appropriate for the version of the Windows CE operating system in the target terminal. The SDK must be installed AFTER any development tools.
• The developer writes a new application and interactively debugs it on the target platform or emulator to ensure that it performs as expected.
• The developer distributes the application using one of the methods below.

2.2.1 Application & Data Transfer Methods

The application can be distributed to the terminal via one of the following methods:
• Use the Remote File Viewer to transfer the application as shown in the embedded Visual Tools V4.0 example below in Section 2.3.4 Programming Example - eMbedded Visual C++ v4.0.
• Use the Smart Device Authentication Utility in Microsoft Visual Studio.NET 2003 to deploy copies of the application. An example using Visual Studio.NET 2003 is shown in Section 2.4.3 Sample Microsoft Visual Studio.NET 2003 development session.
• Use Build/Deploy Solution in whichever version of Microsoft Visual Studio is being used, to deploy copies of the application.
• Use the Remote Manager to upload the application using a Web interface. (see section 3.9 Remote Manager Utility)
• Define shared network folders (see section 1.3.5 Accessing Shared Directories) and copy the application from the shared folder to the target device.
• Use Microsoft ActiveSync to transfer the desired files. See Section 2.5 ActiveSync Reference
• Use removable media such as a Compact Flash card or PCMCIA card.
2.3 Development with Microsoft eMbedded Visual C++ v4.0

Note: Throughout this chapter, when referring to steps performed on the LANpoint PLUS, use the Remote Display Utility where appropriate. Execute the Remote Display Utility as described in Section 1.3.3.2 Remote Display Utility.

2.3.1 Terminal Setup - eMbedded Visual Tools v4.0

In order to communicate with the terminal during the development process, the terminal will have to be configured in order to communicate over the network.

• Set the IP address, subnet mask, gateway, and DNS values. The most direct method to do this is described in Section 1.3.3 Configuring IP Address Properties. By default the terminal ships from the factory and is set to use DHCP.

• Create a file called debugworkstations.txt in the \FLASH directory using the CEPad.exe program, the MonitorCE Utility, LCmdSet Utility, or via the Setup Utility - see Chapter 3 for instructions for these utilities. This text file contains 2 pieces of information. The first piece of information is the IP address of the application development PC. The second piece of information is the port number to be used for development. These two pieces of information are separated by a colon(:). An example would be 172.13.16.1:4000 Each time the terminal boots, the DebugLauncher application will read this file to establish communications with the development PC.

• Reboot the terminal or execute DebugLauncher.exe after setting the IP/port of the development PC.

2.3.2 Installing eMbedded Visual Tools V4.0

• The system requirements for the development computer are documented by Microsoft

Note: Microsoft eMbedded Visual Tools version 4 should not be installed on the same development computer as the previous version 3. If you have version 3 of the tools installed they must be uninstalled prior to proceeding.

• Insert Disk 1 of eMbedded Tools v4.0 into the CD ROM drive on the development computer. Follow the installation instructions provided by Microsoft Corporation to install.
If the installation program does not automatically start, open Windows Explorer and navigate to the CD-ROM drive icon and right-click on the drive icon. In the pop-up menu, select AutoPlay.

- The installation default settings are recommended.

The *readme.txt* file on the eMbedded Tools v4.0 CD ROM includes information on building with eMbedded Visual C++, known problems, samples, and debugging information, PC-link, and ActiveX for Windows CE.

### 2.3.3 Installing the Intelligent Instrumentation SDK

- Install the OS-version-specific LANpoint PLUS SDK (Software Development Kit) for eMbedded Visual Tools 4.0 after completing the installation of the Microsoft eMbedded Tools software on the development computer.

**Note:** Use the same installation directory that you chose for any other SDK tools installed with the Microsoft eMbedded Tools for the installation of the LANpoint PLUS SDK.

- The SDK contains the files listed in the table below.

<table>
<thead>
<tr>
<th>Folder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPmib\</td>
<td>Folder containing definition file(s) for using SNMP management software with the LANpoint PLUS terminal.</td>
</tr>
<tr>
<td>LANpointPLUS_SDK.msi</td>
<td>LANpoint PLUS SDK for CE.NET 4.2 to use with Microsoft development environments.</td>
</tr>
<tr>
<td>LANpointPLUS_5_SDK.msi</td>
<td>LANpoint PLUS SDK for CE 5.0 to use with Microsoft development environments.</td>
</tr>
</tbody>
</table>

- The SDK is available as a CD in starter kits, can be ordered with the purchase of a terminal, or can be downloaded from www.lanpoint.com. Double-click the appropriate `.msi` file to perform the installation.
- If you currently have a version of Microsoft Visual Studio installed on your development machine you may receive a message telling you download some utilities from Microsoft. Follow those instructions if you wish.
• The SDK setup dialog screens below will be shown; they may vary somewhat depending on the version of Microsoft Visual Studio and SDK being used.
Fill-in the appropriate responses where requested.

2.3.4 Programming Example - eMbedded Visual C++ v4.0

The information below is for eMbedded Visual Tools 4.0.
Use *Start->Programs* to launch eMbdedd Visual Tools v4.0. Begin by establishing a connection between the development PC and the LANpoint PLUS terminal.

Locate the *Configure Platform Manager* entry under the *Tools* menu.
When the **Platform Manager** dialog opens, select **LANpoint PLUS** (for CE.NET 4.2 OS models) or **LANpoint PLUS 5** (for CE 5.0 models) and press the **Properties** button to open the **Device Properties** dialog as shown below.

Press the Configure button after selecting TCP/IP Transport for Windows CE in the Transport: pull-down list to open the **TCP/IP Transport Configuration** dialog. Fill in the **Host IP** (development PC) address as well as the port number setting them to the same values that were specified in the `debugworkstations.txt` file (see Section 2.3.1, *Terminal Setup - eMbedded Visual Tools v4.0*). Press the **OK** button, and back in the **Device Properties** dialog, make sure that **Manual Server** is selected in the **Startup Server**: pull down menu. There is nothing to configure with the manual server.
Press the Test button in the Device Properties dialog to connect to the LANpoint PLUS terminal to transfer the application that will be generated. Acknowledge several screens similar to the ones shown above and below.
Now a basic "Hello World" application can be easily generated from eMbedded Visual Tools v4.0. First, back in the main window, select File->New. A New dialog box will open. In the Projects tab, make selections as shown below. Press the OK button.

Note: Developers should select ARMV4 as the CPU for Windows CE.NET 4.2 models, ARMV4i for Windows CE 5.0 models.
When the *WCE Application* dialog is displayed, pick the *A typical “Hello World” application* radio button and press the *Finish* button. Press *OK* in the *New Project Information* dialog box.
Select Project->Settings. The Project Settings dialog will open. Select the sample HelloWorld project, making sure that the proper platform is chosen - Win32 (WECE ARMV4) Debug for Windows CE.NET 4.2 models, Win32 (WECE ARMV4i) Debug for Windows CE 5.0 models. On the Debug tab, pick a directory on the LANpoint PLUS where the application will be transferred. In this example, the application will be put in \flash\tests. Press the OK button.
To compile, link, and download the file to the target device, select Build->Rebuild All and press the OK button in the Manual Server - Action dialog box. Note that status is listed at the bottom of the screen.

Upon successful compilation of the code, a connection with the target device will be attempted. If a connection occurs, the executable will be downloaded to the device automatically. Below is a Windows Explorer screen on the terminal (via the Remote Display Utility - see Section 1.3.3.2 Remote Display Utility), showing the HelloWorld program at the location chosen.
2.4 Development with Microsoft Visual Studio.NET 2003, Visual Studio 2005 or Visual Studio 2008

A simple example using the C# programming language is shown below.

**Note:** Throughout this chapter, when referring to steps performed on the LANpoint PLUS, use the Remote Display Utility where appropriate. Execute the Remote Display Utility as described in Section 1.3.3.2 Remote Display Utility.

2.4.1 Installing Microsoft Visual Studio

The system requirements for supporting the version of Microsoft Visual Studio being used should be reviewed in order to ensure that the development PC meets the minimum acceptable requirement.

Insert the installation disk for Microsoft Visual Studio into the CD ROM drive on the development computer. Follow the installation instructions provided by Microsoft Corporation to install.

If the installation program does not automatically start, open Windows Explorer and navigate to the CD-ROM drive icon and right-click on the drive icon. In the pop-up menu, select AutoPlay. The installation default settings are recommended.

2.4.2 Installing the LANpoint PLUS SDK

Follow the instructions in Section 2.3.3 Installing the Intelligent Instrumentation SDK.

2.4.3 Sample Microsoft Visual Studio.NET 2003 development session

The section below shows the major steps to develop a simple application and download it to the terminal directly from Visual Studio, one of the several methods that can be used to download programs to the LANpoint PLUS terminal. This example is specific to Microsoft Visual Studio.NET 2003.

Run the device portion of the utility on the LANpoint PLUS terminal by executing \Windows\SDAuthUtilDevice.exe as shown below (via the Remote Display Utility - see Section 1.3.3.2 Remote Display Utility). The IP address of the terminal will be shown in the Available IP Addresses list. Note the address and click the Start button.
Run Microsoft Visual Studio.NET 2003 on the development PC and start the development PC portion of the utility by selecting Tools->Smart Device Authentication Utility. When the Smart Device Authentication Utility dialog opens, fill-in Device IP address with the IP address of the LANpoint PLUS terminal and press the Set up device button. A success message will display when the terminal is ready to accept the application program. Press OK to acknowledge and close the success message. When done, press the Close button to close the Smart Device Authentication Utility dialog.

When the Close button is pressed in the Smart Device Authentication Utility dialog on the development PC, the Authentication Utility on the terminal will also close.

Back in the main Visual Studio.Net 2003 window, press the New Project button in the Projects folder in order to begin the development of a custom application program. A New Project dialog will open (shown as an insert superimposed on screen shot below). Select the language you wish to develop with in the Project Types: window (Visual C# in this example) and pick Smart Device Application in the Templates: window.
Give the application a name (**HelloWorld** in this example) in the **Name:** window, and specify where on the development PC to store the project in the **Location:** window. Press the **OK** button.
The *New Project* dialog will close and the *Smart Device Application Wizard* dialog will open. Pick *Windows CE* as the target platform and pick *Windows Application* as the project type, as shown below. Press the *OK* button.

The application Wizard creates a new project with a default form name, *Form1.cs*, as shown in the design window below.
In this example, the basic skeleton program created by the Wizard will be slightly modified to be a trivial program by adding a label that reads "Hello World".
Once that modification is complete, on the development PC, check and make sure that the transport information is correct for the target LANpoint PLUS terminal.

Select **Tools->Options** to open the **Options** dialog.

In the **Options** dialog, press the **Configure** button to open the **Configure TCP/IP Transport Settings** dialog. Make sure the IP address is correct for the target terminal, then press **OK** to close the dialog, and press **OK** in the **Options** dialog to close it.
Select the location on the terminal where the application executable will be stored by selecting Project->HelloWorld Properties and specifying a location on the terminal that is in non-volatile memory like the path \\flash\tests\ in this example. Press the OK button in the HelloWorld Property Pages dialog when done.
Select **Debug->Start** to build and deploy the application to the terminal and start it in the remote debugger.
On the terminal, you may see some spurious warnings or error messages (via the *Remote Display Utility* - see Section 1.3.3.2 *Remote Display Utility*). If so, acknowledge the message(s).

The terminal will begin execution of the application upon successful deployment.

### 2.5 ActiveSync Reference

Microsoft ActiveSync is a popular transfer mechanism for developers deploying applications to Windows CE devices. This reference section outlines the steps for using ActiveSync with the LANpoint PLUS terminal.

Microsoft ActiveSync can be downloaded from www.microsoft.com. It will be installed on the development PC.

**NOTE:** The features and use of ActiveSync are beyond the scope of this document and users are encouraged to read the documentation that comes as part of the Microsoft ActiveSync product.
2.5.1 Terminal Setup - ActiveSync

On the terminal (via the Remote Display Utility - see Section 1.3.3.2 Remote Display Utility), create a new serial connection. Go to Control Panel and click on the Network and Dial-up Connections icon.

![Network and Dial-up Connections Icon](image)

**Note:** the dialog boxes for making a new connection are a little larger than the display screen. You should change the Settings to Auto Hide the Task bar. One way to do this is to right-click on the task bar and choose Properties. Auto Hide will give you enough room on the display to see the whole dialog box.
In the dialog window, double-click on Make New Connection. In that dialog, name the new connection something like ActiveSync, select Direct Connection and press the Next button. A Device dialog window will open. In the Select a device: pull-down menu, select Serial Cable on COM1 (or COM2 or COM3) and press Configure.
Developing Custom Programs

A *Device Properties* dialog window will open. Verify the connection is 19200 baud, 8 bit, No Parity, 1 stop bit, and Hardware Flow Control. Press **OK**.

![Device Properties dialog](Image)

Back in the *Device* dialog, press the *Finish* button. You will see the *ActiveSync* icon in the *Network and Dial-up Connections* dialog.

![Device dialog](Image)

Go back into *Control Panel* and select the *PC Connection* icon. In the *PC Connection Properties* dialog, change to use the *ActiveSync* entry just created. Then press **OK**.

![PC Connection Properties](Image)
Run the \Flash\CEFlush program to save your new entries in the registry.

Connect a serial cable from COM1 on your PC, to the COM1 port on the LANpoint PLUS terminal. Be sure and use the special ActiveSync cable - see the Microsoft ActiveSync Note box in Section 2.5 ActiveSync Reference.

On the LANpoint PLUS terminal, select Start->Run and prepare to execute a program called repilog. VERY IMPORTANT: Do NOT execute it yet.
On the development PC, double-click on the ActiveSync Icon in the system tray. The Microsoft ActiveSync dialog box is displayed. Select File->Get Connected.

While the program on the PC is trying to connect, execute the repllog program on the LANpoint PLUS terminal.

Occasionally this process will time-out. If so, repeatedly run repllog on the LANpoint PLUS and click Get Connected on the PC until the connection is established, and the connection icon shows in the device system tray.

On the PC, an ActiveSync dialog appears. In the Get Connected dialog, select Next and a New Partnership dialog will open. Select No, to NOT establish a partnership. Then press Next.

A successful connection will display Guest Connected message in the Microsoft ActiveSync dialog box.
2.5.2 Sample Microsoft Visual Studio.NET 2003 Application download via ActiveSync

This example is using ActiveSync with Microsoft Visual Studio.NET 2003, and is similar when used for other download purposes. As noted in other sections of this manual, if you are using a different version of Visual Studio, there will be some variations, but it will be very similar to this example.

Develop your application code as outlined in section 2.4.3 Sample Microsoft Visual Studio.NET 2003 development session at the point where the New Project is created.

- Select Tools->Options and select the Devices entry under Device Tools.
• Change the entry in the *Show devices for platform:* drop down box to Windows CE and make other changes in the drop-down menus, if required, as shown below.
• Press the **Configure** button

![Configure TCP/IP Transport Settings](image)

• Set the IP address value to the IP address of the LANpoint PLUS terminal which is the target device.

• Press the **OK** button

After these steps are performed the project can be built and downloaded to the terminal.
2.6 Uninstalling the SDK

To uninstall the LANpoint PLUS SDK, click on the Change or Remove Programs icon in the Control panel, select LANpoint_PLUS_SDK or LANpoint_PLUS_5_SDK, and press the Remove button. Then answer Yes to the verification prompt.

![Uninstall the LANpoint PLUS SDK](image)

FIGURE 2.1 - Un-Install the LANpoint PLUS SDK

2.7 MSMQ and SQL CE support

2.7.1 Microsoft Message Queue (MSMQ)

Microsoft Message Queuing Service (MSMQ) is included in the Windows CE.NET 4.2 and the Windows CE 5.0 versions of the operating system on the LANpoint PLUS terminal. It lets applications on a server communicate with applications on the LANpoint PLUS terminal quickly and reliably, even when the applications are running at different times or across networks and systems that may be offline at the time messages are sent. Basic application development information is included here for convenience.

Using MSMQ, custom applications communicate by sending and receiving messages that contain text or binary data by means of the MSMQ API and a message queue, without the
applications having to establish or maintain a separate communication conduit. This can be used to create distributed applications without the programming overhead associated with creating and maintaining network connections and persistent data buffering.

The custom application program that uses MSMQ must provide the logic which reads the queue contents, processes the message, and responds as required.

Windows CE MSMQ does NOT support the following:

- Encryption
- Authentication
- MQMail
- Reading from remote queues
- Direct format queue names
- Directory-service interface for public queues

### 2.7.1.1 MSMQ Configuration

The terminal ships with the MSMQ administration program called `/Windows/msmqadm.exe`. This is a protected operating system file - change Windows Explorer’s folder options in order to view it. Tasks administered through MSMQAdm include:

- Browsing local queues
- Purging messages
- Deleting individual messages
- Stopping and starting MSMQ service

To use MSMQ on the unit, you must properly configure the registry in the following ways:

1. Set the device name to something unique on the network (and different from the default device name):
   - In Control Panel, Double-click on the System icon.
   - Select the Device Name folder tab
   - Change the contents of the Device name: window to your chosen device name
   - Press the OK button.

2. Add the start-up entries for MSMQ to the registry:
   - Click the Start button and select Run...
   - In the Run window, type a command such as `\Windows\MSMQAdm.exe dir \Flash\msmq binary`. This configures the system to store the queues on the flash file system - in this example it is in the \Flash\msmq folder - and allow binary queue items, which are required if you are going to post messages to remote queues (on the server, for example).

3. Start the MSMQ service:
   - Click the Start button and select Run...
   - In the Run window, type a command such as `\Windows\MSMQAdm.exe start`
4. Run \Flash\CEFlush.exe to save the registry changes to non-volatile memory.
5. MSMQ can be used immediately (as long as the server is configured correctly - see next section) and will also start automatically when the terminal is rebooted.

### 2.7.1.2 Minimum Server Configuration

The scope of this document does not include detailed configuration of the server side of MSMQ, but the system administrator or someone with technical background should know or be able to access the appropriate instructions for the server platform chosen.

A few guidelines from Microsoft:

On a Windows Server, there are two requirements prior to MSMQ installation:

1. The server machine must be set up as a primary domain controller in order to install Active directory.
2. Active directory must be installed.

The MSMQ server component is then installed using the Control Panel:

- Add/Remove Programs
- Add/Remove Windows Components
- Messaging

It is up to the system administrator or other networking personnel to configure server side MSMQ security/permissions.

### 2.7.1.3 Sample Code

eMbedded Visual C++ sample code can be downloaded from the [www.lanpoint.com](http://www.lanpoint.com) website. This sample eMbedded Visual C++ code, MSMQClient, shows how to create a local queue on the LANpoint PLUS, send a message to a remote queue and delete a local queue.

It also has sample Visual Basic sample code, QueTest. This sample Visual Basic program for the server was created using Microsoft Visual Studio.NET 2003. It gives the user the ability to list public and private queues on the local machine as well as remote machines.

### 2.7.2 SQL CE

SQL CE is factory-installed on Windows CE.NET v4.2 models. It is not a utility per se, but details are included here for convenience. For Windows CE 5.0 models, database support can be loaded as needed - for instance, if SQL Compact 3.5/SQL Server 2005 is desired, you can download it from Microsoft - at the time of publication the url is:

**NOTE:** The following information is applicable to SQL CE only; for other database support, see the appropriate information supplied with the download.

A custom program developer can choose from four options for accessing data residing in a SQL database:

1. **Local database** - create a local database and access it through a custom application or by using the `isqlw20.exe` database tool. The `isqlw20.exe` executable is not included on the LANpoint PLUS, but is included in the Microsoft SQL Server CE 2.0 distribution pack which can be downloaded free of charge from Microsoft.

   This option requires no server side component installation or server configuration.

2. **Remote database via XML** - use the browser and HTTP to send requests/updates via a URL. XML data is returned and displayed in the browser window.

   This option requires no local machine component installation or configuration; all software components and configuration is on the server. It also does not require a client access license (CAL).

3. **Replication** - replicate local database data with another SQL database running on a remote Windows server machine.

   This is the most complicated of the options as it requires a custom application running on the LANpoint PLUS terminal as well as server side component installations and configurations.

4. **Remote Data Access** - directly access tables on the server from the client code.

The choice between the options is up to the system developer and the application requirements. Descriptions of each are given below.

### 2.7.2.1 Using SQL CE - Local Database

#### 2.7.2.1.1 LANpoint PLUS Configuration

No special LANpoint PLUS configuration is necessary.

#### 2.7.2.1.2 Server Configuration

No Database Server is required.

#### 2.7.2.1.3 Sample Code

Sample code can be downloaded from [www.lanpoint.com](http://www.lanpoint.com) at the *Overview* page for the specific terminal model in-use (Stationary LANpoint PLUS, Vehicle-Mount, or LANpoint TIME).
CreateLocalDB is a Visual Studio.NET 2003 sample program written using C#. This application shows how to create a local SQL CE database.

2.7.2.2 Using SQL CE - Remote Database Via XML

2.7.2.2.1 LANpoint PLUS Configuration

No special configuration needed on the terminal.

2.7.2.2.2 Server Configuration

The scope of this document does not include detailed configuration of the XML server side of SQL CE, but the system administrator or someone with technical background should know or be able to access the appropriate instructions for the server platform chosen.

A few guidelines:

On a Windows 2000 Server or Windows 2000 Advanced Server, installed software should include Microsoft SQL 2000 with service pack 2, SQL XML support/extension for IIS, and the .NET Framework. IIS must be configured appropriately for security/permissions.

2.7.2.2.3 Sample Code

IIS Server-side sample code can be downloaded from www.lanpoint.com at the Overview page for the specific terminal model in-use (Stationary LANpoint PLUS, Vehicle-Mount, or LANpoint TIME).

Various server-side IIS files show how to extract SQL data when the HTTP request is received and parsed. A virtual directory where these files will be installed must be created on the IIS server.

2.7.2.3 Using SQL CE - Database Replication

2.7.2.3.1 LANpoint PLUS Configuration

No special configuration needed on the terminal. A local SQL database must exist for replication with the remote SQL database.

2.7.2.3.2 Server Configuration

The scope of this document does not include detailed configuration of the Server or configuration of the SQL Database to support replication, but the system administrator and Database Administrator, or other appropriate personnel with technical background should know or be able to access the appropriate instructions for the server platform chosen.

A few guidelines:
On a Windows 2000 Server or Windows 2000 Advanced Server, installed software should include Microsoft SQL 2000 with service pack 2, Microsoft SQL Server CE 2.0 add-on pack, and the .NET Framework. IIS must be configured appropriately for security/permissions.

2.7.2.3 Sample Code

eMbedded Visual C++ sample code can be downloaded from http://www.lanpoint.com at the Overview page for the specific terminal model in-use (Stationary LANpoint PLUS, Vehicle-Mount, or LANpoint TIME).

This is a sample program shows how to replicate a local database on the LANpoint PLUS terminal with another database running on a remote server. There is a readme.html file included which provides additional information on how to build and run the application.

Visual Basic sample code can be downloaded from http://www.lanpoint.com at the Overview page for the specific terminal model in-use (Stationary LANpoint PLUS, Vehicle-Mount, or LANpoint TIME).

This application allows a user to execute standard SQL statements against a standard SQL database running on a remote server.

2.7.2.4 Using SQL CE - Remote Data Access

This is the easiest of the four methods for a developer. It requires only knowledge of SQL commands in the development language chosen.

2.7.2.5 Additional SQL CE Reference Material:

One source of additional information needed to configure SQL running on the Windows Database Server is:

Chapter 3  Utility Programs

3.1 Setup Utility

The LANpoint PLUS ships from the factory with a utility program that allows the developer or installer to more easily configure the terminal via the integral keyboard and 2-line text display.

The program LP25_setup.exe executes commands matching many of the commands used by the MonitorCE utility (see Section 3.8, Monitor Program). The main difference is that these commands are issued using the terminal's keypad and LCD display instead of via serial command strings received on the LANpoint PLUS' COM1 port from a development PC.

The Setup Utility is launched by pressing the F16 key on the terminal - press the 2nd key and release, then press the F8 key.

Note: Launch of the Setup Utility via the F16 key is implemented via the Programmable Function Key Utility (see Section 3.5, Programmable Function Key Utility), so it can be changed if so desired - see the PFKConfigCE utility at 3.5.2, Editing a Programmable Function Key File.

When the Setup Utility launches successfully, the text display will look similar to this:

![Image of Setup Utility initial screen]

After a short time, the Setup Utility will display its command prompt, Setup>

At the Setup> prompt, the following commands can be used:

<table>
<thead>
<tr>
<th>Parameter Description</th>
<th>Command Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit the Setup Utility:</td>
<td>EXIT</td>
</tr>
</tbody>
</table>
## Utility Programs

### Run Terminal Emulation install/uninstall program:
See Section 3.13.1, *Text-based Terminal Emulation* for more information

<table>
<thead>
<tr>
<th>Parameter Description</th>
<th>Command Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run <em>Terminal Emulation</em> install/uninstall program:</td>
<td>INSTALLTE</td>
</tr>
</tbody>
</table>

Note: nnn.nnn.nnn.nnn represents the new parameter value for the IP address, Subnet mask, Gateway address, or DebugWorkstation address. For WINS and DNS: nnn.nnn.nnn.nnn is new first entry and yyy.yyy.yyy.yyy is new second entry (if desired).

### Remote Display:
Set IP address of the host PC:
View IP address of the host PC:
Launch *Remote Display* client program (*cerdisp.exe*):
Set screen refresh time, in ms:
View screen refresh time, in ms:
See Section 3.2, *Remote Display Utility* for more information

<table>
<thead>
<tr>
<th>Parameter Description</th>
<th>Command Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Address:</td>
<td>RDHOSTnnn.nnn.nnn.nnn</td>
</tr>
<tr>
<td></td>
<td>RDHOSTR</td>
</tr>
<tr>
<td>Gateway Address:</td>
<td>RDSTART</td>
</tr>
<tr>
<td></td>
<td>RDREFRESHxxx where xxx = number of milliseconds</td>
</tr>
<tr>
<td></td>
<td>RDREFRESHR</td>
</tr>
<tr>
<td>Subnet Mask Address:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>WINs Address:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>DHCP Enable Flag:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>DNS Address:</td>
<td></td>
</tr>
</tbody>
</table>

### Parameter Description

<table>
<thead>
<tr>
<th>Viewing:</th>
<th>Changing:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRI</td>
<td>NInnn.nnn.nnn.nnn</td>
</tr>
<tr>
<td>NRG</td>
<td>NGnnn.nnn.nnn.nnn</td>
</tr>
<tr>
<td>NRS</td>
<td>NSnnn.nnn.nnn.nnn</td>
</tr>
<tr>
<td>NRW</td>
<td>NWnnn.nnn.nnn.nnn yyyy.yyy.yyy.yyy</td>
</tr>
<tr>
<td>NRP</td>
<td>NPr where n = 1 to enable, n = 0 to disable</td>
</tr>
<tr>
<td>NRD</td>
<td>NDrn nnn.nnn.nnn.nnn yyyy.yyy.yyy.yyy</td>
</tr>
</tbody>
</table>
### Debugging Operations

**IP Address of Host:**
- **Setting:**
  - Creates or modifies the `debugworkstation.txt` file
  - See section 2.3.1, *Terminal Setup - eMbedded Visual Tools v4.0* for more information.
- **Viewing:**
  - DBGnnn.nnn.nnn.nnn[:pppp]
  - DBGR
  - [:pppp] is the optional port number

---

### Run an executable:
- Allows you to run an application in the foreground or in the background (answer y to the subsequent question *Run in Background? (y/n):* in order to run in background)
- **Command Syntax:**
  - RUN path\filename params where *path* is like `\flash\...` and *filename* includes file extension. A space after RUN and before any optional command line parameters is required.

---

### Repeatedly run an executable:
- Store the program executable information in a file called `loadprogram.txt`:
- View the current contents of `loadprogram.txt`:
- Run the program as defined by the current contents of `loadprogram.txt`. Exits the *Setup Utility* after the launch:
  - **NOTE:** the launched executable always runs in the foreground.
- **Command Syntax:**
  - LOAD path\filename params where *path* is like `\flash\...` and *filename* includes file extension. A space after LOAD and before any optional command line parameters is required.
  - VIEWLOAD
  - LAUNCH

---

### Run ActiveSync client program on LANpoint PLUS terminal:
- **Command Syntax:**
  - REPLLOG

---

---

---

---

---
### Utility Programs

#### 3.2 Remote Display Utility

The LANpoint PLUS has a 2 line by 40 character text-only display. This is sufficient for most AutoID and other data collection uses, but may present challenges to the developer.

<table>
<thead>
<tr>
<th>Parameter Description</th>
<th>Command Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start the Remote Manager Utility:</strong>  &lt;br&gt;See Section 3.9, <em>Remote Manager Utility</em> for more information.</td>
<td>WDCE</td>
</tr>
<tr>
<td><strong>Reboot the LANpoint PLUS:</strong></td>
<td>REBOOT</td>
</tr>
<tr>
<td><strong>Save the contents of the registry to non-volatile memory:</strong>  &lt;br&gt;Note: CEFLUSH is not necessary in order to save any Setup Utility command results (such as IP address) in the registry; it is a handy way to run the CEFLUSH.EXE program. See Section 1.3.7, <em>Saving the Contents of the Windows CE Registry</em> for more information.</td>
<td>CEFLUSH</td>
</tr>
<tr>
<td><strong>Real Time Clock:</strong>  &lt;br&gt;Setting Month:  &lt;br&gt;Setting Day of the Month:  &lt;br&gt;Setting Year:  &lt;br&gt;Setting Hours:  &lt;br&gt;Setting Minutes:  &lt;br&gt;Setting Seconds:  &lt;br&gt;Set Real-time Clock:  &lt;br&gt;View Real-time Clock Setting:  &lt;br&gt;Daylight savings time:  &lt;br&gt;Get current time zone:  &lt;br&gt;Setting current time zone:</td>
<td>TIOx, where x = value from 1 - 12. Example: 3 = March  &lt;br&gt;TIDx, where x = value from 1 and 31.  &lt;br&gt;TIYxxxx, where x = year using four digits.  &lt;br&gt;TIHx, where x = value between 0 and 24.  &lt;br&gt;TIMx, where x = value between 0 and 60.  &lt;br&gt;TISx, where x = value between 0 and 60.  &lt;br&gt;TIR  &lt;br&gt;TIV  &lt;br&gt;DSTx, set x to 1 to enable; set x to 0 to disable.  &lt;br&gt;TZG  &lt;br&gt;TZSxyyy zz, where x = + or -, y = number of minutes the desired time zone is from GMT, z = registry index of desired time zone.</td>
</tr>
</tbody>
</table>
To allow the developer to access all of the functions of the operating system, the LANpoint PLUS comes with the **Remote Display Utility**, which allows remote manipulation of the terminal via the familiar Windows Graphical User Interface using a networked Windows PC. This utility has two parts, a client portion (**CERDISP.EXE**) on the LANpoint PLUS, and a host portion (**CERHOST.EXE**) on the PC.

For information on where to get the host **CERHOST.EXE** program and a common example of usage, including the handy Setup Utility commands that make the process extremely quick and simple, see Section 1.3.3, Configuring IP Address Properties.

### 3.3 CEPad Text Editor Utility

The LANpoint PLUS terminal includes a text editor application called **CEPad**. **CEPad** operates similarly to Windows Notepad, enabling you to edit text files on the terminal via the **Remote Display Utility** (see Section 3.2, Remote Display Utility).

The basic **File** menu allows you to create a **New** file, **Open** an existing file, **Save** the file, use **Save As** to save an existing file with a new filename, and **Exit** the application.

### 3.4 Wedge Utility

The WedgeCE utility allows the wedging of COM port data into the keyboard input stream of the active application. This simplifies the process of writing an application that handles bar code input data, or data streams from RS-232 devices attached to the terminal, by making the data from those sources appear as though it was typed on the keyboard.

This application can wedge data from any or all of the terminal’s serial ports to the keyboard input stream. The **WedgeCE.exe** application is located in the folder \Flash. **Running Multiple Instances of the Wedge Utility**

When the WedgeCE application starts, the application’s icon appears in the system tray (you can see it via the Remote Display Utility - see Section 3.2, Remote Display Utility). Multiple instances of the Wedge application may run simultaneously to handle data from multiple serial ports, displaying an icon for each serial port currently serviced by WedgeCE.

WedgeCE uses COM1 as its default COM port, and uses default communications parameters. To use other COM ports and/or other parameters, create a shortcut (see section 1.3.6, Instructions to Create a Shortcut in Windows CE) to WedgeCE and include the appropriate command line parameters:

**TIP:** To have the shortcut start automatically, place the shortcut file in the \Flash\Startup folder. On Windows CE 5.0 models, you can place the shortcut file in \Windows\Startup folder instead - either location will launch the shortcut automatically upon boot-up.
3.4.1 Configuring WedgeCE Parameters

Application Command Line Format

The format of the application command line is:

```
WedgeCE <COMx> <baud> <parity> <data bits> <stop bits> <delay> <prefix>
```

where:

- `<COMx>` indicates the serial port that WedgeCE will use for its input. The format is COMx, with no spaces, where x has a value of 1, 2 or 3. This parameter is not case-sensitive.
- `<baud>` is the baud rate for the serial port.
- `<parity>` indicates the parity for the serial port. Use n (no parity), o (odd parity) or e (even parity).
- `<data bits>` is the number of data bits for serial communications. For eight data bits, use a value of 8.
- `<stop bits>` is the number of stop bits for serial communications. Use 1, 2, or 15. A value of 15 indicates the use of 1.5 stop bits.
- `<delay>` WedgeCE uses a delay parameter to determine when input from the serial port is complete and the data accumulated should be wedged into the keyboard input stream. A delay parameter of 50 ms is adequate for most bar code scanners. The amount of time required for serial input varies, depending on the type of device that is sending the serial data.
  - If the value in the inter-character delay field is too small, data that should arrive in the keyboard buffer at one time may arrive in several packets containing a few characters.
  - If the value in this field is too large, there may be an excessive delay after the end of the input before the wedging occurs.
- `<prefix>` is a string which is prepended to the data each time a packet is wedged into the keyboard buffer. This parameter is useful in identifying the source of the input data when there are several data sources. The maximum length of the prefix is 32 characters. The special string %d is used once in the prefix string and when wedging occurs, this string is replaced by the COM port number (1, 2 or 3) that received the wedged data. To include a literal % symbol in the prefix, place %% in the prefix string. A single % character is not valid in the prefix string. Only one instance of %d is permitted.

Example

For instance, to use WedgeCE with a serial device that is connected to COM3 and has its communications parameters set to 2400 baud, even parity, 7 data bits, 2 stop bits, you wish to delay a little longer than normal between characters being retrieved by the port, say 100ms, because it is a slow device, and you wish to have "AUTOID:" prepended to the data that is retrieved from the device connected to the COM port, use:

```
WedgeCE COM3 2400 e 7 2 100 AUTOID:
```
Input data will be wedged into the keyboard input stream as AUTOID:input data

If, instead, you wish to know which COM port from which the data is retrieved, use:

    WedgeCE COM3 2400 e 7 2 100 COM%d:
Input data will be wedged into the keyboard input stream as COM3:input data

### 3.4.2 WedgeCE Default Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>COM Port</td>
<td>COM1</td>
</tr>
<tr>
<td>baud</td>
<td>9600</td>
</tr>
<tr>
<td>parity</td>
<td>n (none)</td>
</tr>
<tr>
<td>data bits</td>
<td>8</td>
</tr>
<tr>
<td>stop bits</td>
<td>1</td>
</tr>
<tr>
<td>inter-character delay</td>
<td>50 milliseconds (ms)</td>
</tr>
<tr>
<td>Prefix</td>
<td>None</td>
</tr>
</tbody>
</table>

### 3.5 Programmable Function Key Utility

The LANpoint PLUS terminal has eight physical function keys, F1 - F8. The Programmable Function Key (PFK) utility allows soft keys to be configured to either translate the keystrokes into text strings fed to the keyboard buffer or to launch programs. Using combinations of the key with alt-, ctrl- or shift-, up to 32 functions can be programmed using only the 'F' keys.

Most of the other standard keys can also be assigned different definitions using the PFK utility.

There are two PFK programs that perform two different tasks:

- The *PFKConfig.exe* program is used to create new PFK definitions, edit existing PFK definitions, and delete PFK definitions.
- The PFK program executes the PFK definitions.
3.5.1 Creating a New Programmable Function Key File

The terminal may have one PFK definition file with one or more definitions in it, or multiple definition files with one or more definitions in it, at the discretion of the developer. The PFKConfig dialog is used to create the PFK definition file.

To create a new PFK definition file:

1. Launch the PFKConfigCE.exe application. Do this via the Remote Display Utility (see Section 3.2, Remote Display Utility). The default location for this file is \Flash\.

2. In the PfkConfig - Untitled dialog, click the New button.

3. The New Function key dialog opens.

   In the Function field, press the key or the key combination for the new PFK. A key combination consists of one modifier key (Control, Alt, or Shift) in combination with almost any other key.

   Note: Terminals ship from the factory with a default PFK definition file \flash\default.pfk) that has F16 predefined for launching the Setup Utility. You may want to edit this file instead of creating a new file so that function key definition remains active. Since only one copy of PFK.exe may be executed at a time, you must halt the copy loaded at startup before using a different key definition file.

4. To have the PFK launch an application, select the Launch application radio button and specify the application path and name in the Output field. Command line arguments can be added after inserting a semi-colon - like \xxx.exe;+1 where +1 is the command line parameter.

5. Otherwise, if the PFK hot key is to wedge a string into the keyboard stream, select the Wedge into keyboard stream radio button and type the text string (up to 256 characters long) into the Output field.

   Note: The Caps Lock, Print Screen, and Pause keys may not be used. Multiple modifier keys, control-alt-D for instance, cannot be used on a LANpoint PLUS or LANpoint TIME terminal.

The key or key combination appears in the Function field.

4. To have the PFK launch an application, select the Launch application radio button and specify the application path and name in the Output field. Command line arguments can be added after inserting a semi-colon - like \xxx.exe;+1 where +1 is the command line parameter.

5. Otherwise, if the PFK hot key is to wedge a string into the keyboard stream, select the Wedge into keyboard stream radio button and type the text string (up to 256 characters long) into the Output field.

   Note: To include a literal \ character in the text, type \ in the string. For example, to launch \Flash\Any File.exe when the hot key is pressed, type the text string \Flash\Any File.exe. Failure to do so may result in apparently erroneous messages, such as "unknown key" when trying to edit the .pfk file at a later date, or the graying-out of the "save" button when trying to save the .pfk file. You could alternately use the forward slash (/) character instead of \ in path definitions.
6. Click the OK button. The New Function Key dialog closes and the new function key appears in the list in the PfkConfig-Uncle dialog.

7. To create another PFK, repeat steps 2 through 6.

8. Save the new PFK file by clicking the Save As button. On Windows CE.NET version 4.2 OS models, be sure to save under \Flash\... for permanent storage; on Windows CE version 5.0 OS models, all storage locations are in non-volatile memory and thus permanent.

9. Click the Exit button.

10. Use the LANpoint PLUS terminal’s keyboard or an external keyboard to test the operation of the PFKs, using the instructions in section 3.5.3, Testing a Programmed Function Key.

### 3.5.2 Editing a Programmable Function Key File

1. Run the PFKConfigCE program.

2. In the PFKConfigCE-Uncle dialog, click the Open button and locate the PFK file to edit.

3. Select the desired function key file from the list of functions shown and click the Edit button. The New Function Key dialog opens, displaying the contents of the selected PFK file.

4. Make the desired changes to the PFK file and click the OK button to close the New Function Key dialog.

5. Click the Save button in the PfkConfigCE dialog to save the changes.

6. Click the Exit button to close PFKConfigCE.

7. Test the changes.

### 3.5.3 Testing a Programmed Function Key

**Note:** If the PFK program is running when you edit a PFK file, the changes will not be exhibited immediately because PFK.exe reads the file only on initiation. Stop/restart PFK.exe in this case.

To test programmed key(s):

1. Create a shortcut (see section 1.3.6, Instructions to Create a Shortcut in Windows CE) to PFK.exe and right-click on it.

2. Select Properties from the menu.

3. In the Properties dialog, select the Shortcut tab.

4. In the Target field, enter the path and filename of the function key file to be tested.

   For example:
   
   "\Flash\PFK.EXE"  "\Flash\Folder containing your PFK file\File name.pfk"

5. Click the OK button to exit the dialog.

6. Double-click the shortcut. The PFK icon appears in the system tray. If you do not see this icon in the system tray, repeat steps 2-5 and verify the path is correct.
7. To see which PFKs are currently running on the PFK program, double-click the icon in the system tray. The *Programmable Function Keys* dialog opens, displaying a list of the PFKs in the file specified in the path in step 4.

To close this dialog and have the program continue running, click the *Hide* button.

To close the dialog and shut down this program, click the *Exit* button.

8. Press the key(s) to be tested. Verify that the key functions correctly by noting that the correct program was launched or that text shows-up in an active text window (to test the keys that output text, open a text editor such as *CEPad.exe* before pressing the key).

### 3.5.4 Using a PFK

PFKs are only functional when the *PFK.exe* program is running. To use a PFK, follow the instructions in section 3.5.3, *Testing a Programmed Function Key*. In step 7, click the *Hide* button. The icon for the *PFK.exe* shortcut resides in the system tray when the window is hidden.

### 3.5.5 Deleting a Function Key

1. To delete a function key definition, run *PfkConfigCE.exe*.
2. In the *PfkConfigCE* dialog, click the *Open* button and select the .pfk configuration file.
3. Select the function key definition from the list and click the *Delete* button. The listed entry disappears.

**Note:** If you select the wrong function to delete and have not clicked the *Save* button, you can recover the configuration. Click the *Exit* button and a warning pop-up dialog opens asking *Save changes to YourFileName.pfk?* Click the *No* button to retain the previous settings and close the application.

4. Double-check before clicking the *Save* button.
5. Click the *Exit* button, then stop/restart *PFK.exe* if it is running.

### 3.6 DebugLauncher Utility

Windows CE provides a debug application to simplify the process of creating and debugging applications.

#### 3.6.1 How it Works

There is a shortcut to debuglauncher in the *startup* folder by factory default. DebugLauncher executes *CEMGRC.exe* with the appropriate parameters to establish a platform manager connection to eMbedded Visual Tools 4.0 on the computer at the IP address in the *debugworkstations.txt* file. It also controls the execution of the core components that make connections with the Visual Studio 2005 and 2008 environments.
3.7 Time Synchronization

3.7.1 Time Synchronization for Windows CE.NET 4.2

![Note: The time synchronization utility is ONLY included in Windows CE.NET version 4.2 models.
It is not necessary in Windows CE 5.0 models because the functionality is included in the OS and only needs configuration as to server, synchronization interval, and so on. See configuration instructions in section 3.7.2, Time Synchronization for Windows CE 5.0](http://www.boulder.nist.gov/timefreq/service/time-servers.html)

The synchronization of the LANpoint PLUS’s clock with an external time standard is useful for all types of applications, especially labor tracking and time-and-attendance applications. The terminal supports clock synchronization through the Internet Standard specification RFC 868, also known as the NTP (Network Time Protocol) protocol.

You can find a list of U.S. Government and other time servers at http://www.boulder.nist.gov/timefreq/service/time-servers.html

3.7.1.1 Executing the Time Synchronization Program

The TimeSync program is located in \Flash\TimeSync.exe. Run it via the Remote Display Utility (see Section 3.2, Remote Display Utility).

1. Run TimeSync. The TimeSyncCE dialog will open.
2. Click Settings to open the Settings dialog.
3. Click the Time Server tab.
4. In the Identity field, enter the time server IP address or name.
5. In the Connection field, click the desired radio button setting:
   - TCP: Transmission Control Protocol (default setting)
   - UDP: Universal Data Packet
6. For UDP only: In the UDP Timeout in Seconds field, click the up or down arrow keys to obtain the desired setting. The default setting is 5 seconds.
7. Click the Autorun tab and select one of the radio buttons:
   - Don’t run in background (default)
   - Run every (select values from 0- 59) Hours (radio button) or Minutes (radio button)
   - Run every day at:
     To use this selection, click inside of the box twice, when 12:00:00 appears, enter the desired time using the military time format: Hours(0-23):Minutes(0-59):Seconds(0-59).
To add another time setting, click inside of the box. When more than three time settings are present, scroll bars appear in the box enabling access to a maximum of 32 time settings that can be created.

8. Click the OK button to save the settings and exit the dialog.

9. In the main application window, click the Hide button. This triggers time synchronization and the timesync icon appears in the system tray.

Note: The system clock will be synchronized on the basis of your settings. The time displayed in the system tray may lag system time synchronization but will be synchronized within a few minutes at most.

### 3.7.1.2 Handling Synchronization Failures

Check/adjust the configuration parameters:

1. Double-click the TimeSync icon in the system tray.
2. Click the Retry button. Using military time, set the time to retry every:
   - Hours (select values from 0 - 23)
   - Minutes (select values from 0 - 59)
   - Seconds (select values from 0 - 59)
3. Click the OK button to save the settings and exit the dialog.
4. In the main application window, click the Hide button. The time synchronization icon appears in the system tray.

### 3.7.1.3 Command Line Arguments

The TimeSync program can be launched using the LCmdSet utility or from the command line. The command line will look something like:

\flash\timesync.exe e=1h, h=129.6.15.29

Use one of the two formats for the following command line argument formats. A good use for this is to create a shortcut that is put in the startup folder. See section 1.3.6, Instructions to Create a Shortcut in Windows CE for instructions on creating shortcuts and setting command line parameters.

- \option \argument

or

- \option = \argument

The option may be abbreviated as described in the table below. For example,

\e45
or

e = 45

or

every = 45

The host may be specified as

host 198.182.119.56

or

host = 198.182.119.56

Note: Spaces around the equal sign are ignored. Command line arguments *once*, *every*, and *at* cannot be used together.

<table>
<thead>
<tr>
<th>Timesync.exe Command Line Argument, comma separated</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[t] &lt;time_string&gt;</td>
<td>Configures TimeSync to run every day at a specific time of day. The <em>time_string</em> field uses a 24-hour format (hh:mm:ss), where h = hour, m = minute, and s = second. Up to 32 &quot;at time&quot; arguments can be entered.</td>
</tr>
<tr>
<td>e[very] &lt;interval&gt;m</td>
<td>h</td>
</tr>
<tr>
<td>h[ost] &lt;hostname&gt;</td>
<td>Sets the time-server host. For <em>hostname</em>, use either a valid IP address or host name.</td>
</tr>
<tr>
<td>l[og] &lt;logfile_path_name&gt;</td>
<td>Each time the system clock is synchronized, TimeSync prints a line in the \path\ file <em>logfile_path_name</em>.</td>
</tr>
<tr>
<td>o[nce]</td>
<td>Synchronizes the system clock once and exits.</td>
</tr>
<tr>
<td>p[rotocol] TCP</td>
<td>UDP</td>
</tr>
<tr>
<td>r[etry] &lt;interval&gt;s</td>
<td>m</td>
</tr>
</tbody>
</table>
3.7.2 Time Synchronization for Windows CE 5.0

<table>
<thead>
<tr>
<th>Timesync.exe Command Line Argument, comma separated</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t[imeout] &lt;seconds&gt;</td>
<td>Sets the UDP timeout to seconds.</td>
</tr>
<tr>
<td>-m</td>
<td>suppress the display of the TimeSync dialog window.</td>
</tr>
</tbody>
</table>

**Note:** This section is ONLY for Windows 5.0 models.

For synchronizing time in Windows CE.NET 4.2 models, see section 3.7.1, *Time Synchronization for Windows CE.NET 4.2*

The synchronization of the LANpoint PLUS’s clock with an external time standard is useful for all types of applications, especially labor tracking and time-and-attendance applications. The terminal supports clock synchronization via RFC 2030, also known as the SNTP (Simple Network Time Protocol) protocol.

You can find a list of U.S. Government and other time servers that support SNTP at http://www.boulder.nist.gov/timefreq/service/time-servers.html

### 3.7.2.1 Time Synchronization Configuration Program

Time synchronization functionality is provided by a service running under the Windows CE 5.0 operating system. The properties which control it are stored in the registry. Intelligent Instrumentation has provided a configuration program to eliminate the need to edit the registry directly.

The configuration program is located in `\Flash\SNTPServiceConfig.exe`. Run it via the Remote Display Utility (see Section 3.2, *Remote Display Utility*). Details follow.

1. Run `SNTPServiceConfig`. The `SNTPServiceConfig` dialog will open.
The *Time service status* will generally be "Running". As configured from the factory, it is not given a server with which to synchronize, so no operations will be performed. To configure the server for synchronization operation, use the dialog’s menu:

2. Click on the Service | Properties menu item to open the dialog shown below

3. Set the appropriate server name(s) and/or server IP address(es). The *Servers* item in the dialog is simply an EDIT field to which you can add additional servers, one per line, or from which you can remove servers by simply deleting their corresponding lines.

4. Select the *Update Time from Servers* checkbox.

5. Enter a suitable *Refresh Interval* value of 300000 or greater. This value indicates how often the server(s) will be contacted to synchronize the local clock, in milliseconds. The default value in the application, 600000, corresponds to updating the synchronization every 10 minutes which is usually far too often. More common would be a value like 86400000 to synchronize once per day, or perhaps even larger (86400000 = 60000 milliseconds per minute x 60 minutes per hour x 24 hours per day).

6. Enter a suitable *Max Correction* value. This is the maximum number of milliseconds by which the local clock can be changed to match the server clock. The default value, indicating that any amount of adjustment is reasonable, is usually the correct value to use. If the number of milliseconds to be adjusted is larger than *Max Correction*, the clocks will not be synchronized.

7. Enter the *Recovery Interval after failure* value. Following an unsuccessful attempt to reach the time server(s), this is the number of milliseconds to wait before retrying, with a minimum value of 300000. For wired Ethernet, the default 300000 is usually adequate. For wireless networks, this value may need to be adjusted.

8. Select the *Trust Local Clock* checkbox if you wish to have the local clock treated as at least marginally accurate. When this checkbox is unset and a synchronization operation occurs, the local clock will always be updated with the server's time, no matter what the amount of unsynchronization. When the checkbox is set, the local clock is presumed to be reasonably accurate and the time will only be updated to match the server time when the amount of unsynchronization is less significant than the *Max Correction* value.
9. Press the **OK** button to notify the time synchronization service that the new parameters should now be used and to exit the dialog.

### 3.7.2.1.1 Other pull-down menu items

**Update:** After configuration, the server must be sent an indication that the parameters have been changed. This occurs automatically when you select the **OK** button to accept the new configuration settings. If a setting is changed in some other way, via a registry editor, for example, the **Server | Update** menu item can be used to trigger the server to reload its settings from the registry and begin using them.

**Immediate Sync:** The application provides a means to force an immediate synchronization operation. This is accomplished by choosing the **Service | Sync** menu item. This sends a message to the service telling it to contact a time server and perform a synchronization operation. Note that this synchronization operation forces the local clock to be updated, regardless of the settings for **Trust Local Clock** and **Max Correction**. It also unschedules the next synchronization time and recalculates the next sync time based on the time of the forced sync.

**Immediate Set:** The application provides a means to force an immediate clock operation. This is accomplished by choosing the **Service | Set** menu item. This sends a message to the service telling it to contact a time server and perform a synchronization operation but, unlike the Sync operation, it does not force the local clock to be updated, but uses the **Trust Local Clock** and **Max Correction** settings to determine whether the local clock should be updated or not.

**Start/Stop:** The application can be used to control the service, starting or stopping it using the **Service | Start** and **Service | Stop** menu items. There is seldom any reason to do this, however.

### 3.8 Monitor Program

The Monitor program is a utility that provides one of the ways to configure the LANpoint PLUS’ Ethernet addresses and other parameters via the RS232 port COM1.

The MonitorCE program automatically executes on the terminal at start-up. When the MonitorCE program is running, it monitors COM1 for commands. Any serial communication program, such as HyperTerminal, can be used to communicate with the MonitorCE program.

When another application running on the terminal attempts to open COM1, the MonitorCE program closes. To access the MonitorCE program again, restart the terminal.

### 3.8.1 Required Tools

To use MonitorCE to configure a terminal, you need the following items:
• Null modem serial cable to connect the terminal to a computer, or Intelligent Instrumentation part number SP-600C094.
• A computer with an RS-232 port and a serial communications program, such as HyperTerminal.

Connect the null modem cable to the COM1 port on the terminal and connect the other end of the cable to any available serial port on the computer.

### 3.8.2 Configuring the Communications Program

Run a serial communications program, such as HyperTerminal, on the computer and configure it as follows:

- **Baud rate:** 9600
- **Data bits:** 8
- **Parity:** None
- **Stop bits:** 1
- **Flow control:** none

Instructions for using HyperTerminal follow in the next section. If using another communications program, follow the manufacturer’s instructions.

1. Run HyperTerminal (usually named Hypertrm.exe).
2. In the **Connection Description** dialog, in the name field, type a session name. You can use any name you like, preferably one you will remember at a later date should you need to run MonitorCE again. This dialog also lets you select an icon if you wish. Click the **OK** button.
3. In the **Connect To** dialog, verify that the COMx port to which you connected the cable is selected and click the **OK** button. The COMx Properties dialog opens.
4. In the **Port Settings** tabbed dialog, enter the settings provided above, and click the **OK** button.
5. In the named **Connection** window, open the **File** menu and select **Properties**.
6. Select the **Settings** tab.
7. In the **Emulation** pull-down window, select **ANSI**
8. Click the **ASCII Setup** button.
9. In the **ASCII Setup** dialog, in the **ASCII Sending** section of the dialog, select the checkboxes for **Send line ends with line feeds** and **Echo typed characters locally**.
10. In the **ASCII Receiving** section of the dialog, select the checkbox for **Append line feeds to incoming line ends** and click the **OK** button.
11. Click the **OK** button on the connection properties dialog.
12. When you close the **New Connection** window, a pop-up dialog asks if you want to save the new session that you named. Click the **Yes** button.
### 3.8.3 Starting a MonitorCE Program Session

With the serial communications program running on the computer, power-up the LANpoint PLUS terminal and then start a MonitorCE program session:

1. On the PC in the HyperTerminal (or other) communications program window, type the special string, `@@@@` and press the `Enter` key.
   
   In response, the terminal sends a message to HyperTerminal that reads: MonitorCE Activated.

   **Handy trick:** If you do not see 'MonitorCE Activated', it may be because the serial communications program has not reset the PC’s COM port. In this case, you will see the word "autodetect" at the bottom of the communication program’s window. Force the COMM settings by disconnecting the cable from the terminal and with a paperclip or other metal, connect the cable pins 2 and 3, then repeat step 1. "autodetect" will change to "9600 N-8-1" and when you reconnect the cable to the terminal, it should operate as expected.

2. In the serial communication program window, to display help information, type `h` and press the `Enter` key.

3. The terminal responds with the current command set.

   **Note:** The backspace key is not supported in the MonitorCE program. If the backspace key is used to correct a mistake, it has no effect. One way to get around this is to simply type the rest of the characters as gibberish, then enter the command again after the command is rejected.

### 3.8.4 MonitorCE Program Commands

TABLE 3.1, LANpoint PLUS Parameters contains the MonitorCE program command syntax to use when the serial communications program is running. You can use the MonitorCE program to view or change the parameters in the following table. Reboot the terminal to use any new network settings.

<table>
<thead>
<tr>
<th>Parameter Description</th>
<th>Command Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note: nnn.nnn.nnn.nnn represents the new parameter value for the IP address, Subnet mask, Gateway address, or DebugWorkstation address. For WINS and DNS: nnn.nnn.nnn.nnn is new first entry and yyy.yyy.yyy.yyy is new second entry (if required).</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 3.1 LANpoint PLUS Parameters**
### Parameter Description | Command Syntax
--- | ---
**Remote Display:**  
Set IP address of the host PC: RDHOSTnnn.nnn.nnn.nnn  
View IP address of host PC: RDHOSTR  
Launch Remote Display client program (`cerdisp.exe`): RDSTART  
Set screen refresh time, in ms: RDREFRESHxxx where xxx = number of milliseconds  
View screen refresh time, in ms: RDREFRESHR  
See Section 3.2, *Remote Display Utility* for more information  

**IP Address:**  
Viewing: NRI  
Changing: NInnn.nnn.nnn.nnn

**Gateway Address:**  
Viewing: NRG  
Changing: NGnnn.nnn.nnn.nnn

**Subnet Mask Address:**  
Viewing: NRS  
Changing: NSnnn.nnn.nnn.nnn

**WINS Address:**  
Viewing: NRW  
Changing: NWnnn.nnn.nnn.nnn yyy.yyy.yyy

**DHCP Enable Flag:**  
Viewing: NRP  
Changing: NPn where n = 1 to enable, n = 0 to disable

**DNS Address:**  
Viewing: NRD  
Changing: NDnnn.nnn.nnn.nnn yyy.yyy.yyy.yyy

*TABLE 3.1* LANpoint PLUS Parameters (Continued)
3.9 Remote Manager Utility

The Remote Manager utility provides basic terminal management functionality, including access to the file system, application management, remote reboot capability, and security.
settings. The Remote Manager is a set of Web pages served-up by a Web server on the terminal, called WebDevice, to a browser on a remote computer attached to the network. WebDevice has server-side scripting capability that presents the content of the Remote Manager pages.

**Note:** WebDevice consumes resources and may affect the response time of other applications running on the terminal.

**Note:** The terminal’s IP address must be set before you can use Remote Manager. For information on setting the IP address see Chapter 1, *LANpoint PLUS: Setup & System Configuration*, section 1.3.3.3, *Other Methods to Configure the TCP/IP Properties*.

A terminal administrator who knows the user names and passwords for the LANpoint PLUS terminals can perform the following tasks remotely through a Web browser:

- Assigning the terminal a unique name, a description, and location.
- Performing file management on the terminal: uploading, downloading, copying, and deleting files, creating new directories and deleting existing ones.
- Managing applications running on the terminal, including viewing a list of applications currently executing, launching applications, and terminating applications.
- Rebooting the terminal.
- Controlling the security settings, user name and password, for the above capabilities.

### 3.9.1 Enabling WebDevice for Remote Management

If WebDevice is running on the terminal, you will see the WebDevice icon in the system tray. If not, run \Flash\webdevice\WDCE.exe.

**Note:** Each time WebDevice launches, it creates a text file named *IPaddr.txt* for reference. This file contains the terminal’s IP address and is located in the root directory.

Double-clicking on the WebDevice icon in the system tray via the *Remote Display Utility* (see Section 3.2, *Remote Display Utility*) causes the WebDevice window to open. The window contains four buttons that perform the following tasks:

- **Hide** Closes the dialog window, leaving WebDevice running.
- **Options** Opens a dialog displaying the port setting. The default port number is 80. Only one session of WebDevice can run on any given port.
- **Close** Closes WebDevice and removes the WebDevice icon from the system tray, disabling Remote Manager capabilities.
Clear

Clears the text in the WebDevice message window.

**Password Protection**

Two sections of the Remote Manager are not password protected: the *Home page* and *Help*. All of the other sections of the Remote Manager are password protected. Attempting to access any of the password-protected sections causes the browser to open a pop-up dialog asking for a user name and password. See section 3.9.2.2, *Security Manager: Allowing Access to Folders* for more information.

**Help Links**

The Remote Manager web site contains context-sensitive help. The help link is located in the upper right-hand corner of every page.

General information about the Remote Manager is presented in this manual. For specifics and additional information, refer to the Remote Manager Help pages.

### 3.9.2 Using the Remote Manager

To use the Remote Manager, run a Web browser on a computer and point the browser to the desired terminal by entering the terminal’s IP address in the browser’s URL window.

The home page shown in FIGURE 3.2, *LANpoint PLUS Remote Manager Home Page*, is the first page shown. All other remote management function pages can be accessed from the home page.

![LANpoint PLUS Remote Manager Home Page](image-url)

**FIGURE 3.2 LANpoint PLUS Remote Manager Home Page**
3.9.2.1 Management

The terminal information section of the home page displays the terminal’s IP address, name, location, and description. The management section provides links to four management sections: File Manager, Application Manager, System Manager, and Security Manager. Viewing or modifying any of these items requires a user name and password. Factory default user names and passwords are shown in the table below.

<table>
<thead>
<tr>
<th>Section</th>
<th>User Name</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal Information (view)</td>
<td>none (home page)</td>
<td>none (home page)</td>
</tr>
<tr>
<td>Terminal Information (modify)</td>
<td>admin</td>
<td>admin</td>
</tr>
<tr>
<td>File Manager (view)</td>
<td>user</td>
<td>user</td>
</tr>
<tr>
<td>File Manager (modify)</td>
<td>admin</td>
<td>admin</td>
</tr>
<tr>
<td>Application Manager</td>
<td>admin</td>
<td>admin</td>
</tr>
<tr>
<td>System Manager</td>
<td>admin</td>
<td>admin</td>
</tr>
<tr>
<td>Security Manager</td>
<td>admin</td>
<td>admin</td>
</tr>
</tbody>
</table>

**TABLE 3.2 Default User Names and Passwords**
Most of the pages are self-explanatory. The Security Manager pages require additional descriptions as described below.

### 3.9.2.2 Security Manager: Allowing Access to Folders

The Security Manager provides a means for setting user names and passwords to allow access to various folders that contain the Remote Manager Web pages. Password protection applies at the folder level, not the individual Web page (file) level. Different user names and passwords can be assigned for each folder, restricting access to the web pages within.

To access the Security Manager:


![FIGURE 3.3 Network User Name and Password dialog](image)

2. Enter the user name admin, password admin, and click the OK button. The Security Management main page will open.

![FIGURE 3.4 Security Management main page](image)

**Note:** It is highly recommended that the default user name and passwords be changed with the Security Manager as soon as possible.
To modify security, click on the Edit/Remove link which opens the web page shown in FIGURE 3.5, *WebDevice Server System Configuration page*.

![WebDevice Server System Configuration](image)

**FIGURE 3.5 WebDevice Server System Configuration page**

When you click the links *Add New Protected Folder* or *Edit*, the pop-up dialog shown in FIGURE 3.6, *WebDevice Server Protected Folder* opens.

If you no longer want a folder to be password protected, click *Remove* on the appropriate folder line. The line item is deleted from the list of protected folders. The folder remains on the terminal, though not protected.

![WebDevice Server Protected Folder](image)

**FIGURE 3.6 WebDevice Server Protected Folder**

### 3.9.2.2.1 Editing the User Name or Password for a Protected Folder

If you are editing an existing folder, the *Folder to Protect* field contains the path to the protected folder.
Utility Programs

From the page shown in FIGURE 3.6, WebDevice Server Protected Folder, change the user name and/or password for the folder, then click the Save Settings button to save the settings and exit this page.

**Note:** Do not change the folder name in the Folder to Protect field.

### 3.9.2.2.2 Adding a New Password Protected Folder

When adding new security to a folder, enter the path to the folder, relative to the terminal’s \flash\webdevice\htroot folder, in the Folder to Protect field. For instance, to add security to a folder \flash\webdevice\htroot\newfolder, type \newfolder in the field. Then enter a user name and a password. Click the Save Settings button to save the settings. Close this page.

### 3.10 SerialSocket Utility

The SerialSocket utility allows a remote computer to access the devices connected to one or more serial ports on the LANpoint PLUS terminal. When this application is executed, it listens on several TCP/IP sockets. When a remote host connects to one of the sockets, the application opens the corresponding serial port and establishes a connection between the socket and the serial port. Any data sent to the socket is sent out the serial port. Data that is received through the serial port is sent to the host over the socket connection.

**How it Operates**

The SerialSocket utility is multi-threaded and creates one thread for each serial port on the terminal. Each thread listens on the corresponding TCP/IP socket and handles the connection and opening of the serial port, processing data to and from the port/socket. Sockets are numbered beginning at 3000, with the socket that corresponds to each COMx port found at 3000+x. For example, the socket number for COM2 is 3002.

The SerialSocket utility is located at \Flash\SerialSocket.exe. There are no command line parameters. When you run SerialSocket, the SerialSocket icon is displayed in the Windows CE system tray.

### 3.10.1 Port Configuration for SerialSocket Utility

To use the SerialSocket utility successfully, first set the communications parameters of the target COM port(s) if necessary. The default parameters on the COM ports are 9600 baud, no parity, 8 data bits, one stop bit, and no flow control. If these need to be changed (or verified), use the LCmdSet Telnet server utility - see the appropriate commands in Section 3.12, LCmdSet Utility.

### 3.10.2 Viewing the Status of the SerialSocket Utility

To view the status of the SerialSocket utility via the Remote Display Utility (see Section 3.2, Remote Display Utility):
1. Double-click the SerialSocket icon in the system tray. The *SerialSocket Status* dialog opens. This dialog displays the status of each serial port on the terminal. The status is either *Serial port open* or *Waiting for connect*.
   - The *Serial port open* status indicates a remote host opened the socket corresponding to the serial port and the SerialSocket utility successfully opened the serial port.
   - The *Waiting for connect* status indicates a remote host has not opened the socket corresponding to the serial port.

2. To close the dialog while the utility continues to run, click the *Hide* button.
3. To end the SerialSocket utility, click the *Terminate* button.
4. To update the status of each serial port in the dialog, click the *Refresh* button.

### 3.10.3 Testing SerialSocket with HyperTerminal

You can use HyperTerminal or another similar program to open a socket connection to SerialSocket and send and receive data through one of the terminal's COM ports.

1. Use LCmdSet or the Control Panel to configure the COM port setting to match the serial device connected to the port as mentioned in 3.10.1, *Port Configuration for SerialSocket Utility*.

2. On a computer with TCP/IP connectivity to the terminal, run *Hyper Terminal* and configure a connection as described in section 3.8.2, *Configuring the Communications Program*, EXCEPT configure the *Connect To* dialog to *TCP/IP (Winsock)*, enter the terminal's IP address in the Host address field, and set the port number to the port number for the desired COM port (e.g., 3002 for COM2).

3. Strings typed into HyperTerminal should show on the device connected to the COM port and/or data entered through the COM port should show in the HyperTerminal window.

### 3.11 SNMP

SNMP (Simple Network Management Protocol) is a set of concepts and protocols for the management of networked devices. The SNMP protocol provides a means for a remote manager program running on an administrator’s computer to ask the LANpoint PLUS terminal for information about its status, and in some cases, control the operation of the terminal. SNMP packets include information such as the name of the terminal, the type and version of operating system, a list of network interfaces and their properties, and information about the quality of communications.

The LANpoint PLUS SDK CD includes the *SNMPMIB* folder which contains two MIB (Management Information Base) files, *iii.mib* and *iii-lpce.mib*, for your use.
3.12 LCmdSet Utility

The LCmdSet application acts as a server for the Telnet protocol. This application facilitates an Ethernet connection between a LANpoint PLUS terminal running the LCmdSet utility and a host computer running a Telnet client application.

3.12.1 Running LCmdSet

The LCmdSet utility, located at `\flash\lcmdset.exe`, does not have a user interface. When this program launches (by factory default, it launches at startup), it begins listening for one TCP connection to the Telnet port. Only one active client at a time may be connected to the terminal via this port. When one client disconnects, another client connection can be accepted.

3.12.1.1 Host Computer: Launching a Telnet Client

To communicate with the LCmdSet utility, the host computer must run a Telnet client utility, such as HyperTerminal:

On a computer with TCP/IP connectivity to the terminal, run Hyper Terminal and configure a connection as described in 3.8.2, Configuring the Communications Program, EXCEPT configure the Connect To dialog to TCP/IP (Winsock), enter the terminal’s IP address in the Host address field, and set the port number to 23.

3.12.2 Issuing Commands

When you attach to LCmdSet in a Telnet session, you will see the remote> prompt. To issue a command, simply type the command followed by any command parameters and press <enter>.

3.12.2.1 Examples

To view the LANpoint PLUS’s IP address: type

```
show ipaddress<enter>
```

The unit will respond with:

```
IP address: nnn.nnn.nnn.nnn (adapter 0)
```

where nnn.nnn.nnn.nnn is the hard-coded IP address of the built-in wired Ethernet adapter of the LANpoint PLUS. For DHCP assigned addresses, use `show assigned addresses` instead.

To set the LANpoint PLUS IP address and any other settable parameters, you must be privileged. Type

```
set privileged<enter>
```

When the LANpoint PLUS responds with
password>

type

system<enter>

You will now be privileged and will be allowed to change the unit’s IP address. To change the
IP address, type

change ipaddress nnn.nnn.nnn.nnn<enter>

where nnn.nnn.nnn.nnn is the new IP address for the unit.

The unit will respond with

 Changed IP address: nnn.nnn.nnn.nnn (adapter 0)

and the new IP address will take effect after the next reboot. This command also turns DHCP off.

3.12.3 Set Commands

<table>
<thead>
<tr>
<th>Set Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET PRIVILEGED</td>
</tr>
<tr>
<td>Instructs the server to allow privileged commands. When the Set Privileged command is received, the client is prompted for the privilege password, which LCmdSet does not echo. If the password matches, the privileged mode is entered. The default password is system.</td>
</tr>
<tr>
<td>SET NOPRIVILEGE</td>
</tr>
<tr>
<td>Instructs the server to return to the unprivileged mode.</td>
</tr>
<tr>
<td>SET ECHO ON</td>
</tr>
<tr>
<td>Overrides the echo setting negotiated between LCmdSet and the client terminal program with respect to echo. Set the echo on or off.</td>
</tr>
</tbody>
</table>

3.12.4 Show Commands

Notes for Show commands:

1. The IP address (ipaddr), subnet mask (mask), and gateway address (gateway) are entered and displayed in dotted decimal notation (127.0.0.1)
2. The optional ADAPTER parameter defaults to zero, which corresponds to the built-in LAN9000-compatible adapter.
3. The values displayed for IP addresses, the subnet mask, DHCP, and DNS and WINS lists may not be the operational settings. The settings shown are stored in the registry.

<table>
<thead>
<tr>
<th>Show Command Function Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHOW IPADDRESS [ADAPTER n]</td>
</tr>
<tr>
<td>Displays the hardcoded IP address of the indicated network adapter. If the unit is configured for DHCP, the command fails. The command will fail after changing the DHCP flag during the same session. Use SHOW ASSIGNED ADDRESSES if DHCP is ON.</td>
</tr>
<tr>
<td>SHOW SUBNET MASK [ADAPTER n]</td>
</tr>
<tr>
<td>Displays the hardcoded subnet mask for the indicated network adapter. If the unit is configured for DHCP, the command fails. The command will fail after changing the DHCP flag during the same session. Use SHOW ASSIGNED ADDRESSES if DHCP is ON.</td>
</tr>
<tr>
<td>SHOW GATEWAY [ADAPTER n]</td>
</tr>
<tr>
<td>Displays the hardcoded gateway address for the indicated network adapter. If the unit is configured for DHCP, the command fails. The command will fail after changing the DHCP flag during the same session. Use SHOW ASSIGNED ADDRESSES if DHCP is ON.</td>
</tr>
<tr>
<td>SHOW DHCP [ADAPTER n]</td>
</tr>
<tr>
<td>Displays the status of the DHCP option for the indicated network adapter (ON or OFF).</td>
</tr>
<tr>
<td>SHOW DNSLIST [ADAPTER n]</td>
</tr>
<tr>
<td>Displays the list of DNS addresses configured for the indicated network adapter. Items in the list are separated by semicolons.</td>
</tr>
<tr>
<td>SHOW WINSLIST [ADAPTER n]</td>
</tr>
<tr>
<td>Displays the list of WINS addresses configured for the indicated network adapter. Items in the list are separated by semicolons.</td>
</tr>
<tr>
<td>SHOW ADAPTER NAME[S]</td>
</tr>
<tr>
<td>The singular form of this shows the name of the network adapter which corresponds to the indicated network adapter number. The plural form shows a list of all adapters. This might be necessary to identify which network adapter to use when changing an option for an adapter other than the default.</td>
</tr>
<tr>
<td>SHOW ASSIGNED ADDRESSES [ADAPTER n]</td>
</tr>
<tr>
<td>Displays the DHCP-assigned IP address, subnet mask, gateway address, DNS server list, WINS server list, and DHCP server assigned to the unit by DHCP on the last restart of the unit. If the unit is configured for statically-assigned addresses, the command fails. The command will fail after changing the DHCP flag during the same session. Use SHOW IPADDRESS and other commands like that for statically-assigned addresses.</td>
</tr>
<tr>
<td>SHOW ENETADDRESS [ADAPTER n]</td>
</tr>
<tr>
<td>Displays the MAC address.</td>
</tr>
</tbody>
</table>
### Show Command Function Calls

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHOW SPEED [PORT portnum]</td>
<td>for example: SHOW SPEED PORT 2 to see COM2 baud rate. Displays the baud rate setting stored in the configuration of the indicated serial port. The factory default speed is 9600 baud.</td>
</tr>
<tr>
<td>SHOW PARITY [PORT portnum]</td>
<td>Displays the parity setting stored in the configuration of the indicated serial port. The factory default parity is <em>none</em>.</td>
</tr>
<tr>
<td>SHOW FLOW CONTROL [PORT portnum]</td>
<td>Displays the flow control option (handshaking) in the stored configuration for the indicated serial port. The factory default flow control is <em>none</em>.</td>
</tr>
<tr>
<td>SHOW STOPBITS [PORT portnum]</td>
<td>Displays the number of stop bits in the stored configuration for the indicated serial port. The default is port 1 (COM1). The factory default stopbits is <em>1</em>.</td>
</tr>
<tr>
<td>SHOW CHARSIZE [PORT portnum]</td>
<td>Displays the character size (number of data bits) in the stored configuration for the indicated serial port. The factory default charsize is <em>8</em>.</td>
</tr>
<tr>
<td>SHOW DATETIME</td>
<td>Displays the current local date and time of the unit. The date is displayed in mm/dd/yyyy (month/day/year).</td>
</tr>
<tr>
<td>SHOW TIMEZONE</td>
<td>Displays the time zone of the unit, including the offset from GMT. It also displays whether the unit's time zone is presently observing Daylight Savings Time or Standard Time.</td>
</tr>
<tr>
<td>SHOW PROCESSLIST</td>
<td>Displays a list of the processes running on the unit. Displays the process ID, which can be used to terminate the process using the STOP command, and the exe file of the process.</td>
</tr>
<tr>
<td>SHOW DEBUGWORKSTATIONS</td>
<td>Displays the list of debugworkstations specified in the <em>debugworkstations.txt</em> file in the \Flash directory. Items in the list are separated by semicolons.</td>
</tr>
<tr>
<td>SHOW MEMORY</td>
<td>Displays the available physical and virtual memory</td>
</tr>
<tr>
<td>SHOW PLATFORMNAME</td>
<td>Displays the platform name - for the LANpoint PLUS, this will be <em>LANpoint_PLUS</em> (CE 4.2) or <em>LANpoint_PLUS_5</em> (CE5.0).</td>
</tr>
<tr>
<td>SHOW REMOTE DISPLAY HOST</td>
<td>Displays the IP address of the remote display host PC (see section 3.2, Remote Display Utility).</td>
</tr>
</tbody>
</table>
### 3.12.5 Change Commands

Notes for Change commands:

1. The commands in this section require the client be in the privileged mode.
2. After issuing a change command it takes several seconds for the registry to write to persistent storage, wait for the "remote->" prompt before rebooting the unit.
3. The IP address (ipaddr), subnet mask (mask), and gateway address (gateway) are entered and displayed in dotted decimal notation (127.0.0.1)
4. The optional ADAPTER parameter defaults to zero, which corresponds to the built-in LAN9000-compatible adapter.
5. For any network settings to take affect the unit must be rebooted.

#### Change Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGE IPADDRESS ipaddr [ADAPTER n]</td>
<td>Changes the IP address for a network adapter in the unit. When this command is performed, the DHCP flag for the adapter is automatically set to OFF.</td>
</tr>
<tr>
<td>CHANGE SUBNET mask [ADAPTER n]</td>
<td>Sets the subnet mask for a network adapter in the unit. When this command is performed, the DHCP flag for the adapter is automatically set to OFF.</td>
</tr>
<tr>
<td>CHANGE GATEWAY gateway [ADAPTER n]</td>
<td>Sets the gateway address for a network adapter in the unit. When this command is performed, the DHCP flag for the adapter is automatically set to OFF.</td>
</tr>
<tr>
<td>CHANGE DHCP ON</td>
<td>OFF [ADAPTER n]</td>
</tr>
<tr>
<td>CHANGE DNSLIST iplist [ADAPTER n]</td>
<td>Changes the DNS addresses for a network adapter in the unit. The list can contain multiple IP addresses, separated by semicolons.</td>
</tr>
</tbody>
</table>
### Change Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGE WINSLIST iplist [ADAPTER n]</td>
<td>Changes the WINS addresses for a network adapter in the unit. The list can contain multiple IP addresses, separated by semicolons.</td>
</tr>
<tr>
<td>CHANGE SPEED baudrate [PORT portnum]</td>
<td>Sets the baud rate for the indicated serial port. Port 1 corresponds to COM1. The new configuration is effective the next time an application that uses the settings stored in the registry opens the port. The factory default speed is 9600 baud.</td>
</tr>
<tr>
<td>CHANGE PARITY NONE</td>
<td>EVEN</td>
</tr>
<tr>
<td>CHANGE FLOW CONTROL NONE</td>
<td>XONXOFF</td>
</tr>
<tr>
<td>CHANGE STOPBITS 1</td>
<td>15</td>
</tr>
<tr>
<td>CHANGE CHARSIZE 7</td>
<td>8 [PORT portnum]</td>
</tr>
<tr>
<td>CHANGE DATETIME mm/dd/yyyy hh:mm:ss</td>
<td>Sets the date and local time of the unit using the format month, day, year, hour, minute, second.</td>
</tr>
<tr>
<td>CHANGE TIMEZONE gmtOffset [DST]</td>
<td>Sets the time zone of the unit. The gmtOffset specifies the offset of the unit's time zone, in hours, from GMT. The offset may be fractional (such as 7.5), and/or negative (such as -7). If the DST flag is set to 1, the unit's time zone is presumed to be operating in Daylight Savings Time. If the flag is set to 0, Standard Time is assumed. If there are multiple time zones at the offset specified in the command, the server will display a list of matching time zones for selection.</td>
</tr>
<tr>
<td>CHANGE PRIVPASS</td>
<td>Attempts to change the privileged password for the unit. LCmdSet prompts for the new password and saves it to the registry.</td>
</tr>
<tr>
<td>CHANGE REGISTRY [REMOVEACTIVE</td>
<td>REMOVEALL</td>
</tr>
</tbody>
</table>
### Change Commands

<table>
<thead>
<tr>
<th>Change Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGE DEBUGWORKSTATIONS wslist</td>
<td>Changes the contents of the <code>debugworkstations.txt</code> file in the <code>\Flash</code> directory to include the specified list of workstation IP addresses and TCP/IP port numbers. The format is <code>nnn.nnn.nnn.nnn[:port]</code> where the port is optional and the default is 5000. The list can contain multiple items separated by semicolons. Note: If you are using Windows XP on your development computer, pick a different port number, such as 4500 or 5002. 5000 is reserved for a different function in WinXP.</td>
</tr>
<tr>
<td>CHANGE REMOTE DISPLAY HOST</td>
<td>Changes the IP address of the remote display host PC (see section 3.2, Remote Display Utility).</td>
</tr>
<tr>
<td>CHANGE REMOTE DISPLAY REFRESHTIME</td>
<td>Changes the number of milliseconds between refreshes of the remote display contents (see section 3.2, Remote Display Utility). Values &gt; 1999999998 are negative numbers. A negative number indicates no scheduled refresh.</td>
</tr>
</tbody>
</table>

### 3.12.6 Services Commands

The following services commands are available in CE 5.0 and later models. All of the commands in this section except LIST require the client be in the privileged mode.

<table>
<thead>
<tr>
<th>Services Commands</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICES LIST - This command lists all of the services currently loaded and indicates the current state of each.</td>
<td></td>
</tr>
<tr>
<td>SERVICES START serviceinstance - This command starts the indicated service instance. The serviceinstance is a string of the format <code>&lt;prefix&gt;&lt;number&gt;;</code>, such as <code>BTS1:</code></td>
<td></td>
</tr>
<tr>
<td>SERVICES STOP serviceinstance - This command stops the indicated service instance. The serviceinstance is a string of the format <code>&lt;prefix&gt;&lt;number&gt;;</code>, such as <code>BTS1:</code></td>
<td></td>
</tr>
<tr>
<td>SERVICES LOAD servicename - This command activates an inactive service. The servicename is the friendly service name from the registry, <code>HTTPD</code>, for example</td>
<td></td>
</tr>
<tr>
<td>SERVICES UNLOAD serviceinstance - This command deactivates an active service. The serviceinstance is a string of the format <code>&lt;prefix&gt;&lt;number&gt;;</code>, such as <code>BTS1:</code></td>
<td></td>
</tr>
<tr>
<td>SERVICES REFRESH serviceinstance - This command notifies the indicated service instance that its registry parameters have changed and that it should reload them and begin using the new settings. The serviceinstance is a string of the format <code>&lt;prefix&gt;&lt;number&gt;;</code>, such as <code>BTS1:</code></td>
<td></td>
</tr>
</tbody>
</table>
3.12.7 Operational Commands

All of the commands in this section except QUIT require the client be in the privileged mode.

<table>
<thead>
<tr>
<th>Operational Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUIT</td>
</tr>
<tr>
<td>Instructs LCmdSet to close the connection to the client and return to the idle state, waiting for a new client to connect.</td>
</tr>
<tr>
<td>REBOOT</td>
</tr>
<tr>
<td>Instructs the unit to be rebooted.</td>
</tr>
<tr>
<td>RUN program [commandline]</td>
</tr>
<tr>
<td>Instructs LCmdSet to execute the indicated program on the unit, passing the specified command line, if any.</td>
</tr>
<tr>
<td>STOP pid</td>
</tr>
<tr>
<td>Instructs the LCmdSet to terminate the indicated program on the unit. The pid is retrieved from the list of running processes via the SHOW PROCESSLIST command.</td>
</tr>
<tr>
<td>TERMINATE</td>
</tr>
<tr>
<td>Instructs the LCmdSet utility to close.</td>
</tr>
</tbody>
</table>

3.13 Thin Client: Terminal Emulation

The LANpoint PLUS terminal can be run as a thin client or standalone with custom application software. Thin client architectures allow the same applications to run on diverse types of hardware, regardless of the operating system and processor selection. For organizations wanting more flexible deployment of applications and to more easily control management costs,
thin client architecture offers an important enhancement to the traditional client-server architecture that is based on PC’s, servers and mainframe computers. LANpoint PLUS thin-client software includes several text based thin clients.

Text based clients include VT/ANSI/HP terminal emulation, IBM5250 terminal emulation, and IBM3270 terminal emulation and are produced by Connect, Inc.

### 3.13.1 Text-based Terminal Emulation

With the Connect text-based thin client running, the LANpoint PLUS terminal appears to the server as a dumb terminal. All application execution, processing, and data storage occur on the server, midrange-computer, or mainframe computer as it would with a native "dumb" terminal. You can install the version you wish - emulating ASCII terminals such as VT100, VT220, and HP92 terminals, emulating IBM 5250 terminals, or emulating IBM3270 terminals.

You can install and run this client-side software at any time, in evaluation mode - it will have an extra "nuisance" message and will time-out every 30 minutes, but it is a full-featured install in every other respect. The assignment of an authorization number, keyed to the terminal’s Ethernet Address, removes these limitations for production use.

#### 3.13.1.1 Text-Based Terminal Emulation Installation Instructions

Before you start, you may want to note the terminal’s hardware address (MAC) address and contact apps_eng@lanpoint.com, or call Intelligent Instrumentation Application Engineering for an authorization code. You will need the Sales Order number from your purchase of the terminal emulation software, or the terminal emulation software serial number that is shown on the "QuickStart Guide" received with each license purchase.

1. Set terminal's IP address and install the Terminal Emulation Software

   - set the terminal's IP address according to instructions found in section 1.3.3, Configuring IP Address Properties.
   - Use the INSTALLTE command in the Remote Display Utility (see Section 3.2, Remote Display Utility) which runs the program \flash\twinclient\install.exe. An install window with several buttons will be displayed. Click on the button to install the terminal emulation type of your choice, or press the corresponding Function key (F1 = VT/HP; F2 = 3270; F3 = 5250, F4 = remove emulation software, etc). After you verify that you wish to install the software, a command window will open where you will see the appropriate files copied as necessary. When done, a success message will be displayed in the install window (i.e. "5250 Emulation software installed").
   - reboot the terminal by re-powering or pressing the F6 key to reboot - you will see a screen that shows "TwinClient TN xx". This is the initial terminal emulation screen.

Note: see section 3.13.1.2, Text-Based Terminal Emulation Un-Install Instructions for exiting the Terminal Emulation software
2. Terminal Emulation Configuration - assign Host Computer IP addresses

- at the "TwinClient TN xx" screen, press shift-C on the terminal's keyboard. This takes you into the emulator's "Edit Functions" configuration menu.
- highlight "Edit Host List" and hit enter. You will see a default host name (this is a valid host belonging to the emulation software’s manufacturer, Connect, Inc. It works with whichever "flavor" of terminal emulation you choose. You can use this for demos if you wish - contact Intelligent Instrumentation’s Technical Support for a valid user ID and password) and the cursor will be positioned for you to enter your host computer's IP address. Enter this value and hit enter.
- At the prompt for Port (number), enter the port number (usually 23 for Telnet sessions) and press F3 to save the changes.
- arrow down to the "Run Emulator" selection and hit enter to exit the configuration menu.
- reboot the terminal for the IP addresses and other menu configuration items to take effect.
- When the "TwinClient TNxx" screen is shown, simply press the Enter key to attach to your host.

Note: These are minimal instructions. Full instructions can be found in the downloadable manual "855M532, LANpoint PLUS QuickStart Guide" at www.lanpoint.com at the Overview page for the specific terminal model in-use (Stationary LANpoint PLUS, Vehicle-Mount, or LANpoint TIME). That manual goes into much more detail with instructions to use DNS, DHCP, multiple (failover) hosts, and advanced configuration utilities.

3.13.1.2 Text-Based Terminal Emulation Un-Install Instructions

Use the INSTALLTE command in the Remote Display Utility (see Section 3.2, Remote Display Utility) which runs the program \flash\twinc\install.exe. An install window with several buttons will be displayed.

Choose the "Remove Terminal Emulation" button (or F4 key) selection. When done, reboot the terminal.
Page Intentionally Blank
This chapter contains information about the APIs (Application Programming Interfaces) for the LANpoint PLUS terminal. APIs are provided for the languages Visual C++4.0, C# and Visual Basic, but not all APIs are available for all languages. *Microsoft eMbedded Visual Tools 4.0* includes embedded Visual C++4.0. *Microsoft Visual Studio.NET 2003, 2005 and 2008* include many languages, including C++, C# and Visual Basic.

Function calls are shown in this chapter, as well as error codes. Download sample code for the APIs from Intelligent Instrumentation’s web site at www.lanpoint.com at the *Overview* page for the specific terminal model in-use (Stationary LANpoint PLUS, Vehicle-Mount, or LANpoint TIME).

### 4.1 Overview

Five APIs are provided:

- Digital I/O
- Time Synchronization (CE version 4.2 only)
- Display
- Display Subclass
- Reboot

Each API includes function calls for C++. Most functions are also available for C# and Visual Basic, though some cannot be provided for various reasons.

**Digital I/O API**

The digital I/O API enables the configuration, monitoring, and control of digital inputs and outputs - eight digital input channels and eight output channels.

**Time Synchronization API (Windows CE.NET version 4.2 models only)**

This API enables you to write your own time synchronization program. This API is not available for C# or Visual Basic. It is not available on Windows CE version 5.0 models because time synchronization is built into the OS.

**Display API**

This API enables you to display suitable text at defined positions of the 2-line by 40 character display, and handle other display functions such as cursor control and custom character mapping.
Display Subclass API

This API is a special interface for connecting a user-defined segment of the text display to a standard Windows EDIT control. This interface adds special behavior to a given EDIT control such as a text box. The subclassing routine does not interfere with the basic operation of the EDIT control and the access to the text display is transparent to the application, other than arranging for the subclassing operation to occur.

This makes the 2-line display easy to handle. For instance, displaying characters and controlling the cursor position on the text display as characters are typed into the EDIT control.

Reboot API

This API enables you to reboot the LANpoint PLUS terminal via the application.

4.1.1 API Header Files and Import Libraries

The LANpoint PLUS SDK installed in Chapter 2, Developing Custom Programs, contains the header files for each of these APIs. Include the header file for each API in your program and link the appropriate import library with your program when using the eMbedded Visual C++ function calls. Function declaration files are provided for use with Visual Basic and Visual C# API function calls.

4.2 Digital I/O API

This digital I/O API enables the configuration, monitoring, and control of digital inputs and outputs.

4.2.1 Input and Output Channels

The digital I/O component provides eight digital input channels and eight output channels. Digital input channels can be used as event counters for signals with frequencies up to 50 Hz. The valid channels for inputs and outputs are shown below.

4.2.1.1 Input Channels

The input channels can be configured to:

- Detect changes in their input states and to inform a program of the changes.
- Return the current state of the input.
- Count low-to-high transitions of the input signals.

<table>
<thead>
<tr>
<th>Number of Input Channels</th>
<th>Valid Channel Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0, 1, 2, 3, 4, 5, 6, 7</td>
</tr>
</tbody>
</table>
4.2.1.2 Output Channels

The output channels are set through software to the desired state:
• Either continuously on or continuously off.
• Configured to pulse the output closed for a short, programmable period of time. This type of setting is useful for an access control application to temporarily unlock a door, and for applications requiring an indicator light or buzzer for a momentary period of time.

<table>
<thead>
<tr>
<th>Number of Output Channels</th>
<th>Valid Channels Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0, 1, 2, 3, 4, 5, 6, 7</td>
</tr>
</tbody>
</table>

4.2.2 Required Files

The programming functions are defined in the header file LP4DIO.h (LP4DIO.cs for C#, LP4DIO.vb for VB), which contains data constant definitions and API function prototypes. The library file LP4DIO.LIB provides linkage between the C/C++ application program and the API functions.

4.2.3 Return Values

A return value of NO_ERROR_LPCE indicates the function completed successfully. A non-zero return value indicates a failure of the requested functionality. Different non-zero numbers indicate specific errors. Numbers greater than zero represent fatal errors and numbers less than zero represent warnings. Error descriptions are located in Section 4.2.6, Digital I/O Error Codes.

4.2.3.1 When Errors Occur

Errors indicate the hardware failed to respond to a given request.

If a critical error occurs:
• Examine the source code to determine if function parameters are causing the error.
• Verify the Digital I/O hardware is properly installed.
• Contact technical support.

4.2.4 Digital I/O Input Functions

There are two output functions and six input functions described below. Each includes the name of the function, a brief description, the parameter list, and a return value.
The digital input functions are:

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LpCEReadBit</td>
<td>Reads the state of one input bit</td>
</tr>
<tr>
<td>LpCEReadByte</td>
<td>Reads the state of all eight input bits</td>
</tr>
<tr>
<td>LpCEConfigureInputBit</td>
<td>Configures the bit for a requested mode</td>
</tr>
<tr>
<td>LpCEGetInputBitMode</td>
<td>Queries the system for the current configuration mode of a bit</td>
</tr>
<tr>
<td>LpCEStoreConfig</td>
<td>Stores the present input modes and present output states of all output channels to</td>
</tr>
<tr>
<td></td>
<td>EEPROM on the digital I/O board</td>
</tr>
<tr>
<td>LpCERetrieveConfig</td>
<td>Reads the EEPROM on the digital I/O board and configures the input modes and output</td>
</tr>
<tr>
<td></td>
<td>states using the most recent settings stored to the EEPROM</td>
</tr>
<tr>
<td>LpCEReadCounter</td>
<td>Returns data from the requested counter bit and optionally resets counts to zero after</td>
</tr>
<tr>
<td></td>
<td>returning data</td>
</tr>
<tr>
<td>LpCETriggerFunction</td>
<td>Sets up a call back to the provided function when an input bit is activated</td>
</tr>
</tbody>
</table>

### 4.2.4.1 Return the State of a Single Channel

The current digital state (high, value 1, or low, value 0) of any input channel bit can be read by this function. The digital state of a bit is valid, regardless of the configuration - it applies even if the bit is configured as a counter. Regardless of whether the bit was previously configured as BIT_DIGITAL or BIT_COUNTER, the data value reflects the voltage at the digital input pin.

To obtain the count of transitions when a channel is configured as BIT_COUNTER, use the LpCEReadCounter(...) function instead.
4.2.4.2 Return the Current State of All Input Channels

The lower eight bits of data return a digital representation of the current states of all eight input channels. The digital states of all bits are valid, regardless of the configuration. This applies even if the bit is configured as a counter, the current state of the input, on or off, is returned.

To obtain the count of transitions when a channel is configured as BIT_COUNTER, use the LpCEReadCounter(...) function instead.

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*data</td>
<td>Returned data value is 0-255.</td>
</tr>
</tbody>
</table>
Function call in eMbedded Visual C++
int LpCEReadByte(int *data);

Function call in C#
int LpCEReadByte(ref int data);

Function call in Visual Basic
LpCEReadByte(ByRef data As Integer) As Integer

4.2.4.3 Configure the Mode of an Input Channel
One of three modes can be assigned to each input channel (bit) - digital (on/off), trigger, or counter.

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit</td>
<td>0 to 7 bit position in digital input type. Configurable this bit by using one of the three modes.</td>
</tr>
<tr>
<td>mode</td>
<td>Use one of these modes:</td>
</tr>
<tr>
<td></td>
<td>BIT_DIGITAL</td>
</tr>
<tr>
<td></td>
<td>BIT_TRIGGER</td>
</tr>
<tr>
<td></td>
<td>BIT_COUNTER</td>
</tr>
</tbody>
</table>

Return value error code:
0   NO_ERROR_LPCE  Command successfully completed.
40  ERROR_INVALID_READ_NULL_BYTE  data parameter is a null pointer.
41  ERROR_W_BYTE_1  Critical error: Command byte write timeout.
42  ERROR_W_BYTE_2  Critical error: Data byte read timeout.

Parameter Description

- **bit**
  - 0 to 7 bit position in digital input type.
  - Configure this bit by using one of the three modes.

- **mode**
  - Use one of these modes:
    - BIT_DIGITAL
      - Normal input query i.e., On (1) or Off (0). See the example for LpCEReadBit.
    - BIT_TRIGGER
      - Change of state from high to low or low to high causes a call-back. See LpCETriggerFunction.
    - BIT_COUNTER
      - bit counts low to high transitions. See the example for LpCEReadCounter.

Return value error code:
0   NO_ERROR_LPCE  Command successfully completed.
50  ERROR_INVALID_CONFIG_BIT  Parameter bit must be between 0 and 7.
51  ERROR_INVALID_CONFIG_MODE  One of three modes are acceptable.

Function call in eMbedded Visual C++
int LpCEConfigureInputBit(int bit, int mode)
4.2.4.4 Obtain the Current Configuration Mode of a Channel

To obtain the current configuration mode of an input channel, use this call to query the system. Each channel operates in one mode at any given time. Other threads and processes that use this API may have affected the mode.

The factory-default power-up setting for all channels is BIT_DIGITAL mode. Subsequent power-up settings may be different, if modified by the LpCEStoreConfig function.

### Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit</td>
<td>bit to return mode data.</td>
</tr>
<tr>
<td>*mode</td>
<td>Returns the last configured mode. mode returns BIT_DIGITAL, BIT_TRIGGER, or BIT_COUNTER depending on the most recent configuration calls.</td>
</tr>
</tbody>
</table>

Return value error code:

- 0   NO_ERROR_LPCE  Command successfully completed.
- 30  ERROR_INVALID_READ_BIT  Parameter bit must be between 0 and 7.
- 31  ERROR_INVALID_READ_NULL_BIT  Data parameter is a null pointer.

### Function call in eMbedded Visual C++

```c++
int LpCEGetInputBitMode(int bit, int *mode);
```

### Function call in C#

```csharp
int LpCEGetInputBitMode(int bit, ref LP4DIOMode mode);
```

### Function call in Visual Basic

```visualbasic
LpCEGetInputBitMode Lib (ByVal bit As Integer, ByRef mode As LP4DIOMode) As Integer
```

4.2.4.5 Store Input Modes and Output States

Execute LpCEStoreConfig to store the present input modes and present output states of all output bits to the EEPROM on the digital I/O board. When the system reboots, the digital I/O sub-system reads and configures the input modes and output states from those stored values.
LANpoint PLUS APIs

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return value error code:</td>
<td></td>
</tr>
<tr>
<td>0     NO_ERROR_LPCE</td>
<td>Command successfully completed.</td>
</tr>
<tr>
<td>90    ERROR_SC_W1</td>
<td>Timeout writing SAVE command.</td>
</tr>
<tr>
<td>91    ERROR_SC_W2</td>
<td>Timeout writing ReadStatus command.</td>
</tr>
<tr>
<td>92    ERROR_SC_R1</td>
<td>Timeout reading status byte.</td>
</tr>
<tr>
<td>93    ERROR_EEPROM_WRITE_PROTECTED</td>
<td>EEPROM presumed write-protected.</td>
</tr>
</tbody>
</table>

Function call in eMbedded Visual C++

```c
int LpCEStoreConfig(void);
```

Function call in C#

```c
int LpCEStoreConfig();
```

Function call in Visual Basic

```
LpCEStoreConfig() As Integer
```

4.2.4.6 Retrieve Input Modes and Output States

The digital I/O sub-system reads the EEPROM on the digital I/O board and configures the input modes and output states using the most recent settings stored by the `LpCEStoreConfig` function. Use `LpCERetrieveConfig` to return the values stored.

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return value error code:</td>
<td></td>
</tr>
<tr>
<td>0     NO_ERROR_LPCE</td>
<td>Command successfully completed.</td>
</tr>
<tr>
<td>100   ERROR_RC_W1</td>
<td>Timeout writing SAVE command.</td>
</tr>
<tr>
<td>101   ERROR_RC_W2</td>
<td>Timeout writing ReadStatus command.</td>
</tr>
<tr>
<td>102   ERROR_RC_R1</td>
<td>Timeout reading status byte.</td>
</tr>
<tr>
<td>103   ERROR_RC_TRIGS</td>
<td>Timeout reading hardware setup.</td>
</tr>
</tbody>
</table>

Function call in eMbedded Visual C++

```c
int LpCERetrieveConfig(void);
```

Function call in C#

```c
int LpCERetrieveConfig();
```
Function call in Visual Basic
LpCERetrieveConfig () As Integer

4.2.4.7 Return Data from a Counter Bit

This call returns the data from the requested counter bit and optionally resets counts to zero after returning the data. Configure the bit as BIT_COUNTER before using this function.

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit</td>
<td>bit to read, 0 thru 7.</td>
</tr>
<tr>
<td>*data</td>
<td>Return data.</td>
</tr>
<tr>
<td>reset</td>
<td>1 resets the counter to zero after read.</td>
</tr>
<tr>
<td></td>
<td>0 leaves the counter value unchanged after this read operation.</td>
</tr>
</tbody>
</table>

Return value error code:
- 0  NO_ERROR_LPCE  Command successfully completed.
- 60 ERROR_INVALID_CONT_BIT  Parameter bit must be between 0 and 7.
- 61 ERROR_INVALID_CONT_NULL  Parameter data must not be NULL.
- 62 ERROR_INVALID_CONT_MODE  bit property must be properly configured before read as a counter.
- 63 ERROR_INVALID_CONT_OVER  Counts rolled over. If input is at the maximum rate of 50 Hz, overrun will occur in 2.6 years.
- 64 ERROR_INVALID_CONT_RESET  Parameter reset must be 0 or 1.

Function call in eMbedded Visual C++
int LpCEReadCounter(int bit, long *data, int reset);

Function call in C#
int LpCEReadCounter(int bit, ref int data, bool reset);

Function call in Visual Basic
LpCEReadCounter
(ByVal bit As Integer, ByRef data As Integer, ByVal reset As Boolean) As Integer

4.2.4.8 Input Bit Transitions: Trigger Activation

This function sets up a call-back to a specified function when one particular bit triggers on a falling edge. This specified application function should perform very minimal processing and terminate its processing as quickly as possible.
Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>bit</em></td>
<td>Bit to trigger function call. Values 0 through 7 associate function f with individual bits.</td>
</tr>
<tr>
<td><em>f</em></td>
<td>Pointer to a function that accepts a long integer parameter and returns a long integer.</td>
</tr>
</tbody>
</table>

Return value error code:
- 0  NO_ERROR_LPCE Command successful.
- 70 ERROR_INVALID_TRIG_BIT Parameter *bit* must be between 0 and 7.
- 72 ERROR_INVALID_TRIG_NULL Invalid function pointer *f*.

Function call in eMbedded Visual C++

```c
int LpCEBitTriggerFunction(int *bit, long (__stdcall *f)(long));
```

Function call in C#

This function is not supported.

Function call in Visual Basic

This function is not supported.

4.2.5 Digital I/O Output Functions

There are two output functions that interact with the output channel(s), determining whether the output(s) are on (voltage equals external voltage that is applied) or off (open).

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LpCEWriteBit</td>
<td>Controls one output state.</td>
</tr>
<tr>
<td>LpCEWriteByte</td>
<td>Controls all eight output states.</td>
</tr>
</tbody>
</table>

4.2.5.1 Control a Single Output Channel

This function provides control over one output channel at a time. Specify the *bit* (from bit 0 thru bit 7) to activate the corresponding output. The *moment* parameter specifies the amount of time the output is activated.

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
For instance,

In the function LpCEWriteBit(2, 0):
  The output signal associated with bit 2 is turned off and remains off until you make another call and redefine the moment.

In the function LpCEWriteBit(7, 24):
  The output channel associated with bit 7 is turned on for 2400 ms, then is turned off.

In the function LpCEWriteBit(5, 255):
  The output channel associated with bit 5 is turned on and remains on until you make another call and redefine the moment.

Return value error code:
0   NO_ERROR_LPCE Command successfully completed.
10  ERROR_INVALID_WRITE_BIT Parameter bit must be 0 thru 7.
11  ERROR_INVALID_WRITE_MOMENT Parameter moment must be between 0 and 255.
12  ERROR_W_BIT_1 Critical error: Command byte write timeout.
13  ERROR_W_BIT_2 Critical error: Moment byte write timeout.

Note: The state of the output may change when using the LpCEWriteByte function.

Function call in eMbedded Visual C++
  int LpCEWriteBit(int bit, int moment);

Function call in C#
  int LpCEWriteBit(int bit, int moment);

Function call in Visual Basic
  LpCEWriteBit (ByVal bit As Integer, ByVal moment As Integer) As Integer
4.2.5.2 Turn All Outputs on or off

This function uses the lower eight bits of the parameter data to turn all outputs on or off. To set a bit for momentary output, use the LpCEWriteBit function.

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Data bit pattern is output to the 8-bit data output port.</td>
</tr>
</tbody>
</table>

Return value error code:
- 0 NO_ERROR_LPCE: Command successfully completed.
- 20 ERROR_INVALID_WRITE_BYTE: Parameter data must be between 0 and 255.
- 21 ERROR_W_BYTE_1: Critical error: Command byte write timeout.
- 22 ERROR_W_BYTE_2: Critical error: Data byte write timeout.

Function call in eMbedded Visual C++
```
int LpCEWriteByte(int data);
```

Function call in C#
```
int LpCEWriteByte(int data);
```

Function call in Visual Basic
```
LpCEWriteByte (ByVal data As Integer) As Integer
```

4.2.6 Digital I/O Error Codes

This table contains all of the Digital I/O API error codes.

<table>
<thead>
<tr>
<th>Error Value</th>
<th>Return Value Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to All Functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>NO_ERROR_LPCE</td>
<td>Command successfully completed.</td>
</tr>
<tr>
<td>Output Functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>ERROR_INVALID_WRITE_BIT</td>
<td>Parameter bit must be between 0 and 7.</td>
</tr>
<tr>
<td>11</td>
<td>ERROR_INVALID_WRITE_MOMENT</td>
<td>Parameter moment must be between 0 and 255.</td>
</tr>
<tr>
<td>12</td>
<td>ERROR_W_BIT_1</td>
<td>Critical error: Command write timeout.</td>
</tr>
<tr>
<td>Error Value</td>
<td>Return Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>13</td>
<td>ERROR_W_BIT_2</td>
<td>Critical error: Moment bit write timeout.</td>
</tr>
<tr>
<td>20</td>
<td>ERROR_INVALID_WRITE_BYTE</td>
<td>Parameter moment must be between 0 and 255.</td>
</tr>
<tr>
<td>21</td>
<td>ERROR_W_BYTE_1</td>
<td>Critical error: Command byte write timeout.</td>
</tr>
<tr>
<td>22</td>
<td>ERROR_W_BYTE_2</td>
<td>Critical error: Data byte write timeout.</td>
</tr>
</tbody>
</table>

**Input Functions**

<table>
<thead>
<tr>
<th>Error Value</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>ERROR_INVALID_READ_BIT</td>
<td>Parameter bit must be between 0 and 7.</td>
</tr>
<tr>
<td>31</td>
<td>ERROR_INVALID_READ_NULL_BIT</td>
<td><em>data</em> parameter is null pointer.</td>
</tr>
<tr>
<td>32</td>
<td>ERROR_INVALID_READ_MODE</td>
<td>Bit requested was not previously configured as BIT_DIGITAL_CONFIGURED by LpCEConfigureInputBit.</td>
</tr>
<tr>
<td>33</td>
<td>ERROR_R_BIT_1</td>
<td>Critical error: Command bit write timeout.</td>
</tr>
<tr>
<td>34</td>
<td>ERROR_R_BIT_2</td>
<td>Critical error: Data bit read timeout.</td>
</tr>
<tr>
<td>40</td>
<td>ERROR_INVALID_READ_NULL_BYTE</td>
<td><em>data</em> parameter is a null pointer.</td>
</tr>
<tr>
<td>41</td>
<td>ERROR_R_BYTE_1</td>
<td>Critical error: Command byte write timeout.</td>
</tr>
<tr>
<td>42</td>
<td>ERROR_R_BYTE_2</td>
<td>Critical error: Data byte read timeout.</td>
</tr>
<tr>
<td>50</td>
<td>ERROR_INVALID_CONF_BIT</td>
<td>Parameter <em>bit</em> must be between 0 and 7.</td>
</tr>
<tr>
<td>51</td>
<td>ERROR_INVALID_CONF_MODE</td>
<td>Parameter <em>mode</em> is not acceptable.</td>
</tr>
<tr>
<td>60</td>
<td>ERROR_INVALID_CONT_BIT</td>
<td>Parameter <em>bit</em> must be between 0 and 7.</td>
</tr>
<tr>
<td>61</td>
<td>ERROR_INVALID_CONT_NULL</td>
<td>Parameter <em>data</em> must not be NULL.</td>
</tr>
<tr>
<td>62</td>
<td>ERROR_INVALID_CONT_MODE</td>
<td><em>bit</em> property must be properly configured as BIT_COUNTER_CONFIGURED by LpCEConfigureInputBit before it is read as a counter.</td>
</tr>
<tr>
<td>63</td>
<td>ERROR_INVALID_CONT_OVER</td>
<td>Counts rolled over. Given input at the maximum rate of 50 Hz, overrun will occur in 2.6 years.</td>
</tr>
<tr>
<td>64</td>
<td>ERROR_INVALID_CONT_RESET</td>
<td>Parameter <em>reset</em> must be 0 or 1.</td>
</tr>
<tr>
<td>70</td>
<td>ERROR_INVALID_TRIG_BIT</td>
<td>Parameter <em>bit</em> must be between 0 and 7.</td>
</tr>
<tr>
<td>72</td>
<td>ERROR_INVALID_TRIG_NULL</td>
<td>Parameter <em>f</em> is a NULL pointer.</td>
</tr>
</tbody>
</table>
4.3 Time Synchronization API

The Time Synchronization API can be used to include time synchronization in a custom application written in C++ on a Windows CE 4.2 unit only. An NTP time server must be available on the network. The NIST Network Time Service provides several Internet time servers with addresses listed on the web site at: http://www.bldrdoc.gov/timefreq/service/ntc.htm.

The TimeSyncCE.EXE utility can also be used, or used instead. It uses the TS868CE.DLL API and can be configured to periodically update the system clock. This .DLL can use either TCP (Transmission Control Protocol), or UDP (User Datagram Protocol) to communicate with the time server. If UDP is chosen, a UDP timeout value should also be specified. The integral values of TCP and UDP are equal to the constants IPPROTO_TCP and IPPROTO_UDP defined in the file WINSOCK.H.

This API is available for eMbedded C++; it is not supported in C#, Visual Basic, or CE5.0 models.

**Required Library Files**

The dynamic link library and import library file names are:

- TS868CE.DLL
4.3.1 Time Server Functions

There are seven functions for the time synchronization API.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetServer</td>
<td>Specifies a Unicode string containing the network name or IP address of the time server.</td>
</tr>
<tr>
<td>SetProtocol</td>
<td>Sets the identity of the time server.</td>
</tr>
<tr>
<td>SetUDPTimeout</td>
<td>Sets the timeout in seconds for a UDP protocol.</td>
</tr>
<tr>
<td>GetServerTime</td>
<td>Gets the server time.</td>
</tr>
<tr>
<td>GetSystemTime</td>
<td>Gets the time of the local computer’s system clock.</td>
</tr>
<tr>
<td>GetTimeDifference</td>
<td>Calculates the time difference by subtracting the server time from the system time.</td>
</tr>
<tr>
<td>SyncSystemWithServer</td>
<td>Synchronizes the system clock with the clock on the server.</td>
</tr>
</tbody>
</table>

### 4.3.1.1 Network Name or IP Address of the Time Server

**Function Parameters and Return Status Values**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*server</td>
<td>This Unicode string specifies the network name or IP address of the time server.</td>
</tr>
<tr>
<td>Return value error code: 3 RESOLVE_HOSTNAME</td>
<td>Failed to resolve the timer server identity. This error is returned if a name instead of an IP address was specified and the DNS server(s) specified in the Network Control Panel item could not resolve the name, or if DNS servers were not specified. The server can be either in the form of an IP address or a host name, but must not be NULL.</td>
</tr>
</tbody>
</table>
Function call in eMbedded Visual C++
void IISetServer(const TCHAR *server);

4.3.1.2 Set Protocol to use with the Time Server
Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proto</td>
<td>Choose TCP or UDP.</td>
</tr>
</tbody>
</table>

Function call in eMbedded Visual C++:
void IISetProtocol(TSProtocol proto);

4.3.1.3 Set Timeout for a UDP Protocol
Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>seconds</td>
<td>Sets the timeout in seconds (from 1 to 59) when using UDP protocol.</td>
</tr>
</tbody>
</table>

Function call in eMbedded Visual C++
void IISetUDPTimeout(int seconds);

4.3.1.4 Get Server Time
Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>**pTime</td>
<td>Parameter **pTime returns the server time in pTime. Declare a variable of type struct tm * and pass a pointer to this variable. pTime must not be NULL.</td>
</tr>
</tbody>
</table>

Return value error code:
GET_TIME Query to get the server time failed; it timed out.

Function call in eMbedded Visual C++
int IIGetServerTime(struct tm **pTime);

### 4.3.1.5 Return Current LANpoint PLUS System Time

**Function Parameters and Return Status Values**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>pTime</strong></td>
<td>Parameter <strong>pTime</strong> returns the system time (local computer’s system clock) in <em>pTime</em>. Declare a variable of type <code>struct tm *</code> and pass a pointer to this variable. pTime must not be NULL.</td>
</tr>
</tbody>
</table>

**Function call in eMbedded Visual C++**

```c
int IIGetSystemTime(struct tm **pTime);
```

### 4.3.1.6 Calculate Time Difference

This call calculates the difference by subtracting the server time from the system time.

**Function Parameters and Return Status Values**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*pSeconds</td>
<td>The call subtracts the server time from the system time and return the difference in pSeconds. pSeconds must not be NULL.</td>
</tr>
</tbody>
</table>

**Return value error code:**

GET_TIME Query to get the server time failed; it timed out.

**Function call in eMbedded Visual C++**

```c
int IIGetTimeDifference(long *pSeconds);
```

### 4.3.1.7 Synchronize the System Clock with the Server Clock

**Function Parameters and Return Status Values**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>synchronizes the system clock with the clock on the server. This forces a call to GetSystemTime( ) and GetServerTime( )</td>
<td></td>
</tr>
</tbody>
</table>
LANpoint PLUS APIs

Return value error code:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET_TIME</td>
<td>Query to get the server time failed; it timed out.</td>
</tr>
<tr>
<td>SET_TIME</td>
<td>Failed to set the system time with ::SetLocalTime( ).</td>
</tr>
</tbody>
</table>

Function call in eMbedded Visual C++

```c
int IISyncSystemWithServer(long);
```

*pDiff returns the pre-synchronization time difference.

### 4.3.2 Time Sync API Error Codes in eMbedded Visual C++

The descriptions for the time synchronization API error codes are as follows:

<table>
<thead>
<tr>
<th>Return Value Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORKED</td>
<td>No errors.</td>
</tr>
<tr>
<td>CREATE_SOCKET</td>
<td>Failed to create a new socket.</td>
</tr>
<tr>
<td>WINSOCK_INIT</td>
<td>WSASStartup( ) failed to initialize Winsock library.</td>
</tr>
<tr>
<td>RESOLVE_HOSTNAME</td>
<td>Failed to resolve the time server identity.</td>
</tr>
<tr>
<td>CONNECTION</td>
<td>Failed to connect to the socket.</td>
</tr>
<tr>
<td>GET_TIME</td>
<td>Query to get the server time failed due to time-out.</td>
</tr>
<tr>
<td>SET_TIME</td>
<td>Failed to set the system time with ::SetLocalTime( ).</td>
</tr>
<tr>
<td>PRIVILEGE</td>
<td>Inadequate privileges to adjust system time.</td>
</tr>
<tr>
<td>UNK_PROTOCOL</td>
<td>The specified protocol is not supported.</td>
</tr>
<tr>
<td>SELECT</td>
<td>UDP only. Failed to determine socket status.</td>
</tr>
<tr>
<td>SEND_UDP</td>
<td>UDP only. Failed to send time query.</td>
</tr>
</tbody>
</table>

### 4.4 Display API

The Display API enables you to control all aspects of the 2-line by 40-character text display. In general, the functions fall into four categories - cursor movement and control, character output, screen control (clearing portions of the screen, for instance), and special character (custom bitmap dot pattern) generation.

For porting existing applications to the terminal, or easing some LCD handling tasks, also see 4.5, Display Subclass API.
The programming functions are defined in the header file `displayAPI.h` (`displayAPI.cs` for C#; `displayAPI.vb` for VB), which contains data constant definitions and API function prototypes. The library file `DisplayAPI.LIB` provides linkage between the C/C++ application program and the API functions.

### 4.4.1 Display API Functions

There are seventeen Display API functions.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cursorxy</td>
<td>Positions the cursor at a specific character position on the LCD display.</td>
</tr>
<tr>
<td>get_cursorxy</td>
<td>Returns the x and y coordinates of the current cursor position on the LCD display.</td>
</tr>
<tr>
<td>cursorunderlineon</td>
<td>Show or hide the underline cursor.</td>
</tr>
<tr>
<td>get_cursorunderlineon</td>
<td>Returns the current state of the underline cursor.</td>
</tr>
<tr>
<td>cursorblockon</td>
<td>Show or hide the block cursor.</td>
</tr>
<tr>
<td>get_cursorblockon</td>
<td>Returns the current state of the block cursor.</td>
</tr>
<tr>
<td>printxy</td>
<td>Display a string beginning at a specific position on the LCD display.</td>
</tr>
<tr>
<td>wprintxy</td>
<td>Display a string beginning at a specific position on the LCD display, including an (optional) table of Unicode and other custom character bitmaps.</td>
</tr>
<tr>
<td>printcursor</td>
<td>Display a string beginning at the current position of the cursor.</td>
</tr>
<tr>
<td>clrscr</td>
<td>Clear the LCD display and move the cursor to the home position (0,0).</td>
</tr>
<tr>
<td>clrlinesegment</td>
<td>Clear a portion of a line on the LCD display.</td>
</tr>
<tr>
<td>clreol</td>
<td>Clear the characters on the LCD display from the current cursor position to the end of the line.</td>
</tr>
</tbody>
</table>
4.4.1.1 Position the cursor - cursorxy

Relocates the cursor to the given position on the display. This does not affect the cursor type (block/underline), nor does it affect the visibility of the cursor.

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>The x coordinate where the cursor will be placed, 0 to 39.</td>
</tr>
<tr>
<td>y</td>
<td>The y coordinate where the cursor will be placed, 0 or 1</td>
</tr>
</tbody>
</table>

Function call in eMbedded Visual C++

```cpp
void cursorxy(int x, int y);
```

Function call in C#

```csharp
void cursorxy(int x, int y);
```

Function call in Visual Basic

```vbnet
cursorxy (ByVal x As Int32, ByVal y As Int32)
```
4.4.1.2 Get the current position the cursor - get_cursorxy

Returns the cursor x y coordinates on the display. This does not affect the cursor type (block/underline), nor does it affect the visibility of the cursor.

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>The x coordinate where the cursor is currently positioned, 0 to 39.</td>
</tr>
<tr>
<td>y</td>
<td>The y coordinate where the cursor is currently positioned, 0 or 1</td>
</tr>
</tbody>
</table>

Function call in Embedded Visual C++

```cpp
void get_cursorxy(int *x, int *y);
```

Function call in C#

```c#
void get_cursorxy(ref int *x, ref int *y);
```

Function call in Visual Basic

```vbnet
get_cursorxy(ByRef x As Int32, ByRef y As Int32)
```

4.4.1.3 Show or hide the underline cursor

Shows or hides the underline cursor at its current position. The cursor position is not changed by this operation. The flashing block cursor can be enabled/disabled separately from the underline cursor - see 4.4.1.5, Show or hide the block cursor.

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>on</td>
<td>Sets the state (1 = True, 0 =False) of the underline cursor</td>
</tr>
</tbody>
</table>

Function call in Embedded Visual C++

```cpp
void cursorunderlineon(BOOL on);
```

Function call in C#

```c#
void cursorunderlineon(bool on);
```
Function call in Visual Basic

    cursorunderlineon (ByVal on As Boolean)

4.4.1.4 Return the current state of the underline cursor

Returns the current state of the underline cursor (shown or hidden).

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>Returns the state (1 = True, 0 = False) of the underline cursor</td>
</tr>
</tbody>
</table>

Function call in eMbedded Visual C++

    BOOL get_cursorunderlineon();

Function call in C#

    bool get_cursorunderlineon();

Function call in Visual Basic

    get_cursorunderlineon () As Boolean

4.4.1.5 Show or hide the block cursor

Shows or hides the flashing block cursor at its current position. The cursor position is not changed by this operation. The flashing block cursor can be enabled/disabled separately from the underline cursor - see 4.4.1.3, Show or hide the underline cursor.

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>on</td>
<td>Sets the state (1 = True, 0 = False) of the block cursor</td>
</tr>
</tbody>
</table>

Function call in eMbedded Visual C++

    void cursorblockon(BOOL on);

Function call in C#

    void cursorblockon(bool on);
Function call in Visual Basic

cursorblockon (ByVal on As Boolean)

4.4.1.6 Return the current state of the block cursor

Returns the current state of the block cursor (shown or hidden).

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>Returns the state (1 = True, 0 = False) of the block cursor</td>
</tr>
</tbody>
</table>

Function call in eMbedded Visual C++

BOOL get_cursorblockon();

Function call in C#

bool get_cursorblockon();

Function call in Visual Basic

get_cursorblockon () As Boolean

4.4.1.7 Display text on the LCD display

Displays an indicated character string at a given screen position.

Character output is the process of actually determining what character values are to be displayed by the LCD. The vast majority of the characters are fixed, both in their dot pattern and in the character code, which causes them to be displayed.

Character output is accomplished by sending an ASCII string to the API, along with a location where the first character in the string is to be displayed. When character output occurs, characters that would be placed 'off-screen', are lost. That is, if ten characters are to be displayed, starting at (x,y) of (35,0), then only the first five will actually be written to the display, in character positions 35, 36, 37, 38, and 39 on the first row of the display. The other five will simply be ignored by the API; there is no harm in sending them.

Display of characters affects cursor position, so if the cursor is to remain at its initial location after text display, it must be moved by the application code. When characters are written to the end of a line on the screen, the cursor is set to the last character on the line where the text was written.
The controller on the display provides the ability to create a small number of custom characters (dot patterns) to be created and used. See section 4.4.1.17, Define custom LCD character display shape bitmaps.

The mapping of a given Unicode character to a given dot pattern can be altered from the default mapping used by the LCD. For most printable characters, this is not necessary, as the mapping of the Unicode character relates directly to the dot pattern that represents the Unicode text character. If it is necessary, see sections 4.4.1.18, Unicode character set remapping, 4.5.1.1, Associate an EDIT control with the LCD display, and 4.5.1.2, Associate a checkbox BUTTON control with the LCD display.

**Function Parameters and Return Status Values**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>The x coordinate where the cursor will be placed, 0 to 39.</td>
</tr>
<tr>
<td>y</td>
<td>The y coordinate where the cursor will be placed, 0 or 1</td>
</tr>
<tr>
<td>s</td>
<td>The string to be displayed. The string must be null-terminated, in standard C style; the null character is not displayed.</td>
</tr>
</tbody>
</table>

**Function call in eMbedded Visual C++**

doxygen printxy (int x, int y, char *s);

**Function call in C#**

void printxy (int x, int y, byte[] s);

**Function call in Visual Basic**

printxy (ByVal x As Int32, ByVal y As Int32, ByVal s() As Byte

4.4.1.8 Display text on the LCD display, with optional character mapping

Displays an indicated character string at a given screen position. If the mapping table is specified, the string is scanned and items replaced as indicated by the table prior to display. Note that the original parameter string would be modified, in this case. See also 4.4.1.18, Unicode character set remapping.
Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>The x coordinate where the cursor will be placed, 0 to 39.</td>
</tr>
<tr>
<td>y</td>
<td>The y coordinate where the cursor will be placed, 0 or 1</td>
</tr>
<tr>
<td>s</td>
<td>The string to be displayed. The string must be null-terminated, in standard C style; the null character is not displayed. NOTE that, unlike printxy(), the string passed to wprintxy() is a Unicode string, not an ASCII string.</td>
</tr>
<tr>
<td>mt</td>
<td>Optional table of mappings between Unicode characters which might appear in the string and particular characters in the display character set. This might be used if a special Unicode character did not automatically map to the right display character (ü for example). If no mapping is desired, set this parameter to NULL. See also Section 4.4.1.17, Define custom LCD character display shape bitmaps</td>
</tr>
</tbody>
</table>

**Function call in eMbedded Visual C++**

```c
void wprintxy( int x, int y, TCHAR *s, MAPTABLE *mt );
```

**Function call in C#**

```c
void wprintxy( int x, int y, string s, byte[] mt );
```

**Function call in Visual Basic**

This function is not supported.

### 4.4.1.9 Display a string at current cursor position

Displays an indicated character string at the current cursor position on the display. If the last character to be printed would have been off-screen, the cursor is placed at the last output character location.
Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>The string to be displayed beginning at the current cursor position. The string must be null-terminated, in standard C style; the null character is not displayed.</td>
</tr>
</tbody>
</table>

**Function call in eMbedded Visual C++**

```c
void printcursor(char *s);
```

**Function call in C#**

This function is not supported

**Function call in Visual Basic**

This function is not supported

### 4.4.1.10 Clear the LCD display

Clears (to 'space' characters) the contents of the display and moves the cursor to the home position, (0, 0). The cursor type and visibility state are not affected.

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>

**Function call in eMbedded Visual C++**

```c
void clrscr();
```

**Function call in C#**

```c
void clrscr();
```

**Function call in Visual Basic**

```vbnet
clrscr()
```

### 4.4.1.11 Clear a line on the LCD

Clears (to 'space' characters) the contents of one of the two lines of display. The cursor type and visibility are not affected.
### Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>Line on the LCD on which to clear all the displayed characters, 0 or 1</td>
</tr>
</tbody>
</table>

#### Function call in eMbedded Visual C++

```cpp
void clrline(int line);
```

#### Function call in C#

```csharp
void clrline(int line);
```

#### Function call in Visual Basic

```vbnet
crline(ByVal line As Integer)
```

### 4.4.1.12 Clear a portion of a line on the LCD

Clears (to 'space' characters) a section of a single line of the text display. The cursor type and visibility are not affected.

Note that the range is inclusive of xstart and xend positions on the line - both will be cleared.

### Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xstart</td>
<td>Starting character position for the characters to be cleared, 0 to 39.</td>
</tr>
<tr>
<td>xend</td>
<td>Ending character position for the characters to be cleared, xstart to 39.</td>
</tr>
<tr>
<td>line</td>
<td>Line on the LCD on which to clear the specified characters, 0 or 1</td>
</tr>
</tbody>
</table>

#### Function call in eMbedded Visual C++

```cpp
void clrlinesegment(int xstart, int xend, int line);
```

#### Function call in C#

```csharp
void clrlinesegment(int xstart, int xend, int line);
```
LANpoint PLus APIs

**Function call in Visual Basic**

```vbnet
void clrlesegment(ByVal xstart As Integer, ByVal xend As Integer, ByVal line As Integer)
```

### 4.4.1.13 Clear a line on the LCD, from cursor position to end-of-line

Clears from the current cursor position to the end of the line on which the cursor is presently located. The character 'under' the cursor is cleared as part of this operation.

**Function Parameters and Return Status Values**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>

**Function call in eMbedded Visual C++**

```cpp
void clreol();
```

**Function call in C#**

```csharp
void clreol();
```

**Function call in Visual Basic**

```vbnet
cлёol()
```

### 4.4.1.14 Enable LCD display

Controls the 'power' state of the LCD display. The data is retained when the display is off; when it is set back to on, the previous displayed data will be restored.

**Function Parameters and Return Status Values**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>on</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sets the 'power' state (1 = True, 0 = False) of the LCD display. When TRUE, enables display of characters on the screen. When FALSE, turns off the display so that neither characters, nor cursor, are displayed. The display data is retained when the display is off - when subsequently set to TRUE, the previous display will be restored.</td>
</tr>
</tbody>
</table>

**Function call in eMbedded Visual C++**

```cpp
void displayon(BOOL on);
```
4.4.1.15 Turn LCD backlight on

Turns the display backlight on or off. Changes in the backlight state have no effect on the content of the display. In order to override the automatic on/off control of the backlight, call `backlight auto (false)` before using this call.

**Function Parameters and Return Status Values**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>on</code></td>
<td>Turn the LCD backlight on or off (1 = True, 0 = False)</td>
</tr>
</tbody>
</table>

4.4.1.16 Set LCD backlight to automatic control by OS

Display backlight on/off is controlled by the OS. Changes in the backlight state have no effect on the content of the display. In order to manually control the on/off control of the backlight, call `backlight auto (false)` before using `backlighton()`.

**Function Parameters and Return Status Values**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>on</code></td>
<td>LCD backlight controlled by OS on or off (1 = True, 0 = False)</td>
</tr>
</tbody>
</table>

```csharp
void displayon(bool on);
```

**Function call in Visual Basic**

displayon(ByVal on As Boolean)

```csharp
void backlighton(bool on);
```

**Function call in eMbedded Visual C++**

`void backlighton(BOOL on);`

**Function call in C#**

`void backlighton(bool on);`

**Function call in Visual Basic**

`backlighton(ByVal on As Boolean)`
Function call in C#
void backlightauto(bool on);

Function call in Visual Basic
backlightauto(ByVal on As Boolean)

4.4.1.17 Define custom LCD character display shape bitmaps

This function establishes the bitmap for a custom character in the display character set.

A 5x8 dot pattern is displayed at each location on the LCD display. A given character code is associated with each pattern. The display allows for up to 8 custom dot patterns to be defined in addition to the standard character set it displays. Once a custom dot pattern is defined, it is retained until the display is reset or reprogrammed.

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>The character code whose bitmap is being redefined, 0 to 7.</td>
</tr>
<tr>
<td>map</td>
<td>Specifies the bitmap of the character, an 8-row x 5-column map. The map must contain eight (8) elements, one per row of the character bitmap. There are 5 bits in each entry, one per column of the character bitmap. A zero bit will turn the corresponding pixel 'off'. A non-zero bit will turn the corresponding pixel 'on'.</td>
</tr>
</tbody>
</table>

Function call in eMbedded Visual C++
void mapcharacter( char c, char map[8] );

Function call in C#
void mapcharacter( byte c, byte[] map );

Function call in Visual Basic
mapcharacter(ByVal c As Byte, ByVal map() As Byte);

4.4.1.18 Unicode character set remapping

Character mapping refers to the mapping between Unicode characters which might be sent to the display and the desired display character set items with which they correspond.
For example, the Unicode character ü (Unicode 252), does not exist in the display character set at position 252; it is at position 225 instead. For a given Unicode character, a mapping may be required to display the expected character instead of the ‘default’ LCD character.

This function modifies the input string, mapping Unicode characters called out in the \texttt{mt} parameter to the corresponding display character set characters.

**Function Parameters and Return Status Values**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{s}</td>
<td>The string to be displayed. The string must be null-terminated, in standard C style; the null character is not displayed.</td>
</tr>
<tr>
<td>\texttt{mt}</td>
<td>Mapping table which converts a Unicode character into specific characters in the LCD display’s character set or a custom character bitmap (dot pattern).</td>
</tr>
</tbody>
</table>

**Function call in eMbedded Visual C++**

```c
void wmap(TCHAR *s, MAPTABLE *mt);
```

**Function call in C#**

function not supported in C#

**Function call in Visual Basic**

function not supported in Visual Basic

### 4.5 Display Subclass API

A common user interaction with the text display is editing an input for use by the application program. The Display Subclass APIs are a special interface to allow for connecting a user-defined segment of the text display to a standard Windows EDIT control. This interface depends on the Display API in section 4.4, Display API and uses an operation called Subclassing to add the special behavior to a given EDIT control.

An EDIT control subclassed in this way allows the graphical EDIT control’s operations to be tied by the subclass routine to similar operations on the text display. As characters are added to the EDIT control, for example, the same characters are added to the text display. Cursor movement within the control is also handled. The subclassing routine does not interfere with the basic operations of the EDIT control, so the access to the text display is fairly transparent once the application program is modified to include subclassing.
Likewise a BUTTON control that is configured as a checkbox can be associated with a section of the text display. The subclassing remains in effect until the control is destroyed. When the control gains focus, the current contents will be written to the text display in a defined column range on the given line of the display, along with a small section of text to show the current state of the checkbox ("[X]"). Changes in the state of the checkbox are reflected on the text display by adding or removing the 'X' in the display of the checkbox value.

The programming functions are defined in the header file `displaysubclass.h (displaysubclass.cs for C#)` which contains data constant definitions and API function prototypes. The library file `DisplaySubclass.Lib` provides linkage between the C/C++ application program and the API functions.

### 4.5.1 Display Subclass API Functions

There are two Display SubClass API functions.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextDisplayEDITSubclass</td>
<td>Associates a given EDIT control with a section of the text display.</td>
</tr>
<tr>
<td>TextDisplayBUTTONSubclass</td>
<td>Associates a given BUTTON control, configured as a checkbox, with a section of the text display.</td>
</tr>
</tbody>
</table>

### 4.5.1.1 Associate an EDIT control with the LCD display

`TextDisplayEDITSubclass` subclasses an EDIT control, and associates with it a section of the text display. The subclassing remains in effect until the control is destroyed. When the control gains focus, the current contents will be written to the text display in the xstart to xend range of character columns on the given line of the display. When the EDIT control loses the focus, the specified section of the text display will be cleared to blank. Multiple EDIT controls can use the same section of the text display, if desired.

The `mt` parameter allows specific Unicode characters that might be placed in the EDIT control by the user to be mapped into corresponding characters in the character set of the text display. An example would be mapping a Unicode character from it's natural value to one of the custom user-defined characters in the display - see section 4.4.1.18, *Unicode character set remapping.*

### Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>The window handle of the EDIT control.</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>xstart</td>
<td>Starting character position for the characters to be displayed, 0 to 39.</td>
</tr>
<tr>
<td>xend</td>
<td>Ending character position for the characters to be displayed, xstart to 39.</td>
</tr>
<tr>
<td>y</td>
<td>Line on the LCD on which to display the specified characters, 0 or 1.</td>
</tr>
<tr>
<td>mt</td>
<td>Optional (set to NULL to have none); mapping table which converts the Unicode characters found in the EDIT control into specific characters in the LCD display’s character set (possibly a custom character bitmap).</td>
</tr>
</tbody>
</table>

In C, the mt parameter contains the following:

```c
typedef struct MAPTABLE
{
    int entries;
    MAPENTRY table[1];
} MAPTABLE;
```

The entries field contains the actual number of table entries in the table array. For use with more than one entry, allocate the necessary memory and interpret the pointer as a MAPTABLE pointer. The MAPENTRY item is defined as follows:

```c
typedef struct MAPENTRY
{
    TCHAR unicodeChar;
    unsigned char displayChar;
} MAPENTRY;
```

Return value error code:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Command successfully completed.</td>
</tr>
<tr>
<td>-1</td>
<td>Too many EDIT controls subclassed at the same time. At present, up to 50 EDIT &amp; BUTTON controls, total, can be subclassed.</td>
</tr>
</tbody>
</table>
Function call in eMbedded Visual C++

int TextDisplayEDITSubclass(HWND hwnd, int xstart, int xend, int y, MAPTABLE *mt);

Function call in C#

int TextDisplayEDITSubclass(IntPtr hwnd, int xstart, int xend, int y, MAPENTRY[] mt);

Function call in Visual Basic

This function is not supported

4.5.1.2 Associate a checkbox BUTTON control with the LCD display

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>The window handle of the BUTTON control. If the button is not configured to operate as a checkbox, the call will fail.</td>
</tr>
<tr>
<td>xstart</td>
<td>Starting character position on the LCD display which will be used to mimic the BUTTON control, 0 to 39.</td>
</tr>
<tr>
<td>xend</td>
<td>Ending character position on the text display which will be used to mimic the BUTTON control, xstart to 39.</td>
</tr>
<tr>
<td>y</td>
<td>Line on the LCD on which to display the specified characters, 0 or 1</td>
</tr>
<tr>
<td>mt</td>
<td>Optional (set to NULL to have none) mapping table which converts the Unicode characters found in the BUTTON control into specific characters used on the text display.</td>
</tr>
</tbody>
</table>

Return value error code:

- 0: Command successfully completed.
- -1: Too many BUTTON controls subclassed at the same time. At present, up to 50 EDIT & BUTTON controls, total, can be subclassed.
- -2

Function call in eMbedded Visual C++

int TextDisplayBUTTONSubclass(HWND hwnd, int xstart, int xend, int y, MAPTABLE *mt);
Function call in C#

```csharp
int TextDisplayBUTTONSubclass(IntPtr hwnd, int xstart, int xend, int y, MAPENTRY[] mt);
```

Function call in Visual Basic

```vbnet
TextDisplayBUTTONSubclass(ByVal hwnd As IntPtr, ByVal xstart As Integer, ByVal xend As Integer, ByVal y As Integer, ByVal mt As MAPENTRY) As Integer
```

4.6 Reboot API

The Reboot API enables you to add the capability to reboot the LANpoint PLUS terminal via your application.

The programming functions are defined in the header file ExitWin.h (ExitWin.cs for C#; ExitWin.vb for Visual Basic) which contains data constant definitions and API function prototypes. The library file ExitWin.lib provides linkage between the application program and the API functions for C/C++.

4.6.1 Reboot Calls

There is one Reboot call.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIExitWindows</td>
<td>Reboots the terminal</td>
</tr>
</tbody>
</table>

4.6.1.1 Reboot Function

Function Parameters and Return Status Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIExitWinType</td>
<td>Shutdown type:</td>
</tr>
<tr>
<td></td>
<td>0x00 - Shutdown with extreme prejudice, no messages</td>
</tr>
<tr>
<td></td>
<td>0x01 - Running applications are TOLD that Windows CE is shutting down</td>
</tr>
<tr>
<td></td>
<td>0x03 - Running applications are asked if it's OK to shutdown</td>
</tr>
</tbody>
</table>
Function call in eMbedded Visual C++
    void IIExitWindows(IIExitWinType type);

Function call in C#
    void IIExitWindows(IIExitWinType type);

Function call in Visual Basic
    IIExitWindows( ByVal mode as IIExitWinType )
Index

Numerics
3270 Terminal Emulation - see also Terminal Emulation Software 3-36
5250 Terminal Emulation - see also Terminal Emulation Software 3-36

A
ActiveSync
cable, how to procure 2-24
cable, how to use 2-27
Deployment method 1-11
how to procure 2-24
Setup 2-23
ActiveX 2-5
API
Digital I/O functions 4-2
channel numbering 4-2
error codes 4-12
input functions 4-3
  all channels, returning state of 4-5
  channel, return mode of 4-7
  channel, set mode of 4-6
  counter, returning count optional reset to zero 4-9
input channel, returning state of 4-4
input modes, retrieving 4-8
input modes, storing 4-7
trigger activation 4-9
output functions 4-10
  all channels, controlling 4-12
  output states, retrieving 4-8
  output states, storing 4-7
  single channel, controlling 4-10
return values 4-3
Display functions 4-18
  clear a line on the LCD 4-26
  clear a portion of a line on the LCD 4-27
  clear LCD line from cursor to eol 4-28
  clear the LCD 4-26
  control LCD backlight 4-29
define custom character shapes 4-30
description of 4-18
display text at cursor position 4-25
display text at LCD position 4-23
display text at LCD position, with optional character mapping 4-24
enable LCD display 4-28
files required 4-19
get current cursor position 4-21
get state of block cursor 4-23
get state of underline cursor 4-22
list of 4-19
position the cursor 4-20
show / hide block cursor 4-22
show / hide underline cursor 4-21
unicode character set remapping 4-30
Display subclass
  associate a checkbox BUTTON control to a portion of the LCD display 4-34
  associate an EDIT control with a portion of the LCD Display 4-32
description of 4-31
files required 4-32
list of 4-32
header files and import libraries
  all APIs 4-2
  Digital I/O 4-3
  Display function 4-19
  Display subclass 4-32
  Reboot 4-35
time Synchronization 4-14
overview of all APIs 4-1
Reboot 4-35
  files required 4-35
time Synchronization 4-14
error codes 4-18
files required 4-14
get server time 4-16
set communication protocol 4-16
set time server IP/name 4-15
set UDP timeout 4-16
system clock, get time 4-17
system clock, synchronizing with server clock 4-17
time difference, calculating 4-17
time server function list 4-15
application software
  custom
    bar code input data 3-5
eMbedded Visual C++
      development example 2-7
      installing development environment 2-4
      Microsoft readme file 2-5
eMbedded Visual Tools 4.0 - see also eMbedded Visual C++ 4.0 2-4
file transfer methods 1-11, 2-3
ActiveSync Reference 2-23
general how-to steps 2-3
installing application on LANpoint PLUS/TIME 2-3
selecting the correct software development tools 2-1
Visual Studio
  development example 2-15
  installing development environment 2-15
Auto-ID
devices 1-3
Index

software 1-4

B
backlight, LCD display 1-5
-control via custom program API 4-29
bar code
-wedging data into keyboard input stream 3-5
bar code hardware/software 1-3
baud rate
-configuring with LCmdSet 3-33
reading with LCmdSet 3-31
BUTTON control subclassing 4-31

C
C# - see also application software, custom 2-15
C++, eMbedded - see also Microsoft eMbedded Visual C++ 4.0 2-2
cable
-connections
auxiliary power 1-4
COM (serial) 1-3
external keyboard 1-4
wired Ethernet 1-3
CEMGRC.EXE - see also Debug Launcher 2-9
CEPad text editor utility 3-5
circuit boards 1-2
cpu board 1-2
optional option board 1-2
clock
-see also real-time clock, setting
-with LCmdSet 3-33
-with MonitorCE 3-20
-see also time synchronization 3-11, 3-14
-see also time/date 3-4
COM ports 1-3
default settings 1-3
displaying 3-31
setting 3-33
CompactFlash
-using as additional non-volatile memory 1-3
-using for file transfer 1-11, 2-3
configuring
-Network addresses - see also IP address 1-6
CPU board 1-2
CPU chip 1-2
Cursor control
-Display functions API 4-20, 4-21, 4-22, 4-23
custom programs
-see also - application software, custom 2-1

D
data
-bar code input, wedging 3-5

non-volatile local storage 1-14
database access methods, SQL CE 2-34
Debug Launcher utility 2-9, 3-10
debugworkstations.txt
-configuring with CEPad.exe 2-4
-configuring with LCmdSet 3-34
-configuring with MonitorCE 3-3, 3-20
-example in custom program development session 2-9
-reading with LCmdSet 3-31
development software
-see also - application software, custom 2-1
DHCP, configuring - see also IP address 1-10
Digital I/O - see also API, Digital I/O 4-2
directories, creating shared 1-12
display
-backlight 1-5
-backlight control API 4-29
-custom programming APIs - see also API, display 4-19
custom programming APIs - see also API, display subclass 4-31
-integral text 1-5
dNS address, configuring - see also IP address 1-6
downloading files
-see also file transfer methods 1-11
-via shared directories, how to 1-12

E
EDIT control subclassing 4-31
eMbedded Visual Tools 4.0 - see also application software, custom 2-2
error codes
-Digital I/O API 4-12
-Time Synchronization API 4-18
Ethernet, wired 1-3
troller 1-3
-viewing MAC address with LCmdSet 3-30
-viewing MAC address with MonitorCE 3-20
Ethernet, wireless 1-3
-viewing MAC address with LCmdSet 3-30

F
features
-Auto-ID 1-3
hardware 1-2
circuit boards 1-2
-CPU board 1-2
-Ethernet 1-3
-RAM 1-2
-internal option board
-CompactFLASH slot 1-3
-PCMCIA slot 1-3
wireless Ethernet 1-3
software
  operating system 1-5
utilities 3-1
file transfer methods 1-11, 2-3
via shared directories, how to 1-12
FLASH memory 1-2
flow control (serial)
  configuring with LCmdSet 3-33
  reading with LCmdSet 3-31
function keys, programmable 3-7

G
gateway address
  configuring - see also IP address 1-10

H
hardware
  features 1-2
  see also - features, hardware 1-2
HP Terminal Emulation - see also Terminal
  Emulation Software 3-36
HyperTerminal, testing the SerialSocket utility 3-27
IBM3270 Terminal Emulation - see also Terminal
  Emulation Software 3-36
IBM5250 Terminal Emulation - see also Terminal
  Emulation Software 3-36
installation
  important NOTE concerning keyboard 1-4
installing application software 2-3
IP address
  configuration methods 1-11
  configuring with Control Panel 1-11
  configuring with LCmdSet 3-32
  configuring with MonitorCE 3-2, 3-19
  configuring with Setup Utility & Remote Display
    Utility 1-7
  in debugworkstations.txt file for custom program
    development 2-4
  reading with LCmdSet 3-30
  viewing wired E-NET MAC address with
    MonitorCE 3-20
  viewing wired or wireless MAC address with
    LCmdSet 3-30
wireless configuration
  changing password 1-10
  default password 1-9

K
Keyboard

L
important NOTE concerning keyboard 1-4
keyboard
  external 1-4
  standard integral 1-4
wedging, see also WedgeCE utility 3-6

LANpoint PLUS SDK - see also SDK 2-5
launch a program
  with LCmdSet utility 3-35
  with Remote Manager 3-21
  with Setup Utility 3-3
LCD
  backlight 1-5
  backlight control API 4-29
  custom programming APIs - see also API, display 4-19
  custom programming APIs - see also API, display
    subclass 4-31
LCmdSet Telnet Server utility 3-28
library files
  Digital I/O API 4-3
  Display API 4-19
  Display subclass API 4-32
  Reboot API 4-35
  Time Synchronization API 4-14
load a service in CE5.0
  with LCmdSet utility 3-34

M
memory
  non-volatile CompactFlash slot 1-14
  non-volatile, FLASH 1-2
  RAM 1-2
  viewing available memory with LCmdSet 3-31
Microsoft eMbedded Visual C++ 4.0 - see also
  application software, custom 2-4
Microsoft eMbedded Visual Tools 4.0 - see also
  Microsoft eMbedded Visual C++ 4.0 2-2
Microsoft Message Queue (MSMQ) 2-32
Microsoft Visual Studio - see also application
  software, custom 2-15
MonitorCE serial configuration utility 3-16
  launching 1-6
MSMQ - see also Microsoft Message Queue 2-32

N
net use command syntax - see also file transfer
  methods via shared directories, how to 1-12

O
operating system
  Windows CE version 1-2, 1-5
parity (serial)
  configuring with LCmdSet 3-33
  reading with LCmdSet 3-31
password
  wireless configuration, changing 1-10
  wireless configuration, default 1-9
PC card
  supported types 1-3
  using for file transfer 1-11
PCMCIA card - see also internal option board 1-2
ports, serial RS232 communication 1-3
power-up sequence 1-5
processor, Intel XScale model 1-2
Programmable Function Key - see also PFK 3-7
programs, custom development - see also
  application software, custom 2-4
RAM 1-2
  viewing available memory with LCmdSet 3-31
real-time clock
  see also time synchronization 3-11, 3-14
  see also time/date 3-4, 3-20, 3-33
reboot
  API 4-35
  with LCmdSet utility 3-35
  with MonitorCE utility 3-20
  with Remote Manager 3-21
  with Setup Utility 3-4
registry
  return to factory default 1-14
  saving in non-volatile memory 1-13
service refresh in CE5.0 - with LCmdSet utility 3-34
service register in CE5.0 - with LCmdSet utility 3-35
service unregister in CE5.0 - with LCmdSet utility 3-35
Remote Display utility 3-4
Remote Manager (WebDevice) 1-11, 3-20
sample programs
  eMbedded Visual C++ development example 2-7
  Visual Studio
devlopment example 2-15
SDK
  how to procure 2-5
  installation with eMbedded Visual Tools 4.0 2-5
  installation with Visual Studio 2-15
serial communication (COM) ports 1-3
SerialSocket utility 3-26
Services in CE5.0
  Managing with LCmdSet 3-34
Setup utility 3-1
size of character, in bits (serial)
  configuring with LCmdSet 3-33
  reading with LCmdSet 3-31
SNMP 2-5
software
  APIs 4-1
  custom development
    see also - application software, custom 2-1
    terminal emulation 3-35
  utilities 3-1
software development kit - see also SDK 2-5
SQL CE 2-34
start a service in CE5.0
  with LCmdSet utility 3-34
startup
  creating a startup application shortcut 1-13
  sequence of events 1-5
stop a program
  with LCmdSet 3-35
  with Remote Manager 3-21
stop a service in CE5.0
  with LCmdSet utility 3-34
stop bits (serial)
  configuring with LCmdSet 3-33
  reading with LCmdSet 3-31
subnet mask
  configuring - see also IP address 1-10
TCP/IP
  configuring - see also IP address 1-6
  sockets, SerialSocket utility 3-26
Telnet
  client, launching on a host computer 3-28
terminal emulation software 3-35
terminate a program
  with LCmdSet 3-35
  with Remote Manager 3-21
text editor, CEPad 3-5
thin-client software
  see also, terminal emulation software 3-35
Time Synchronization
  API 4-14
  Windows CE 4.2 3-11
  Windows CE 5.0 3-14
time zone
  reading & setting with MonitorCE 3-20
  reading and setting with Setup Utility 3-4
  reading with LCmdSet 3-31
  setting with LCmdSet 3-33
time/date
  reading & setting with MonitorCE 3-20
  reading and setting with Setup Utility 3-4
  reading with LCmdSet 3-31
  setting with LCmdSet 3-33
TimeSync - see also Time Synchronization 3-11, 3-14

U
unicode character set remapping on LCD display via
  custom program API 4-30
unload a service in CE5.0
  with LCmdSet utility 3-34
uploading files
  see also file transfer methods 1-11
  via shared directories, how to 1-12
utility programs
  CEPad Text Editor 3-5
  Debug Launcher 3-10
  LCmdSet Telnet Server 3-28
  MonitorCE serial configuration utility 3-16
    launching 1-6
  PFK 3-7
  Programmable Function Key - see also PFK 3-7
  Remote Display 3-4
  Remote Manager - see also WebDevice 3-20
  SerialSocket 3-26
  Setup 3-1
  Time Synchronization 3-11, 3-14
  WebDevice 3-20
  WedgeCE 3-5

V
Visual Studio - see also application software,
  custom 2-15
Visual Tools, eMbedded 4.0 - see also Microsoft
  eMbedded Visual C++ 4.0 2-2
VisualBasic - see also application software,
  custom 2-15
VTxxx Terminal Emulation - see also Terminal
  Emulation Software 3-36

W
WebDevice 1-11, 2-3, 3-20
WedgeCE utility 3-5
Windows CE
  operating system 1-5
  registry, saving settings 1-13